The design and implementation of vision-based autonomous rotorcraft landing

by

Andries Matthys de Jager

Thesis presented in partial fulfilment of the requirements for the degree of
Master of Science in Engineering
at Stellenbosch University

Supervisor:
Dr I.K. Peddle
Department Electrical and Electronic Engineering

March 2011

Declaration

By submitting this thesis/dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2011

Copyright © 2011 Stellenbosch University
All rights reserved

Abstract

This thesis presents the design and implementation of all the subsystems required to perform precision autonomous helicopter landings within a low-cost framework.

To obtain high-accuracy state estimates during the landing phase a vision-based approach, with a downwards facing camera on the helicopter and a known landing target, was used. An efficient monocular-view pose estimation algorithm was developed to determine the helicopter's relative position and attitude during the landing phase. This algorithm was analysed and compared to existing algorithms in terms of sensitivity, robustness and runtime.

An augmented kinematic state estimator was developed to combine measurements from low-cost GPS and inertial measurement units with the high accuracy measurements from the camera system. High-level guidance algorithms, capable of performing waypoint navigation and autonomous landings, were developed.

A visual position and attitude measurement (VPAM) node was designed and built to perform the pose estimation and execute the associated algorithms. To increase the node's throughput, a compression scheme is used between the image sensor and the processor to reduce the amount of data that needs to be processed. This reduces processing requirements and allows the entire system to remain on-board with no reliance on radio links. The functionality of the VPAM node was confirmed through a number of practical tests. The node is able to provide measurements of sufficient accuracy for the subsequent systems in the autonomous landing system.

The functionality of the full system was confirmed in a software environment, as well as through testing using a visually augmented hardware-in-the-loop environment.

Opsomming

Hierdie tesis beskryf die ontwikkeling van die substelsels wat vir akkurate outonome helikopter landings benodig word. 'n Onderliggende doel was om al die ontwikkeling binne 'n lae-koste raamwerk te voltooi.

Hoë-akkuraatheid toestande word benodig om akkurate landings te verseker. Hierdie metings is verkry deur middel van 'n optiese stelsel, bestaande uit 'n kamera gemonteer op die helikopter en 'n bekende landingsteiken, te ontwikkel. 'n Doeltreffende mono-visie posisie-en-oriëntasie algoritme is ontwikkel om die helikopter se posisie en oriëntasie, relatief tot die landingsteiken, te bepaal. Hierdie algoritme is deeglik ondersoek en vergelyk met bestaande algoritmes in terme van sensitiwiteit, robuustheid en uitvoertyd.

'n Optimale kinematiese toestandswaarnemer, wat metings van GPS en inersiële sensore kombineer met die metings van die optiese stelsel, is ontwikkel en deur simulasie bevestig. Hoë-vlak leidingsalgoritmes is ontwikkel wat die helikopter in staat stel om punt-tot-punt navigasie en die landingsprosedure uit te voer.

'n Visuele posisie-en-oriëntasie meetnodus is ontwikkel om die mono-visie posisie-en-oriëntasie algoritmes uit te voer. Om die deurset te verhoog is 'n saampersingsalgoritme gebruik wat die hoeveelheid data wat verwerk moet word, te verminder. Dit het die benodigde verwerkingskrag verminder, wat verseker het dat alle verwerking op aanboord stelsels kan geskied. Die meetnodus en mono-visie algoritmes is deur middel van praktiese toetse bevestig en is in staat om metings van voldoende akkuraatheid aan die outonome landingstelsel te verskaf.

Die werking van die volledige stelsel is, deur simulasies in 'n sagteware en hardeware-in-die-lus omgewing,bevestig.

Acknowledgements

I would like to thank the following people:

- Dr Iain Peddle for giving me the freedom to make this project my own, but providing guidance and feedback when required.
- Armscor for providing the funding that made this project possible.
- My family for all the support and motivation.
- All my friends in the ESL for support, advice, chats and plakkies breaks.
- Chris Jaquet for being a great friend and proofreading this thesis.
- Carlo van Schalkwyk and Emile Rossouw for laying the foundations of the current RUAV research platform on which this project is built. Bernard Visser for pioneering the vision-based research in the UAV group.
- Rudi Gaum and Ruan de Hart for providing assistance and advice when needed.
- Wessel Croukamp, Johan Arendse and Ashley Cupido for the technical assistance during the hardware development.

Contents

List of Figures

List of Tables

Nomenclature

Notational Convention

a	Scalar
$x^p,\ y^p$	Scalar component in a pixel frame vector $\mathbf{A}^P$
$x^r,\ y^r,\ z^r$	Scalar component in a real image frame vector $\mathbf{A}^R$
$x^c,\ y^c,\ z^c$	Scalar component in a camera frame vector $\mathbf{A}^C$
$\mathbf{A}$	Vector or matrix
$\tilde{\mathbf{a}}$	Unit vector
$\{\mathbf{A}\}$	Set of vectors
$\mathbf{A}^I$	Vector coordinated in the inertial frame
$\mathbf{A}^B$	Vector coordinated in the body frame
$\mathbf{DCM}_{a\to b}$	Transformation matrix from frame a to frame b
$a(i)$	ith component of $\mathbf{A}$
$a_{x/y}$	x/y component of $\mathbf{A}$
$a_{N/E/D}$	$N/E/D$ component of $\mathbf{A}^I$
$a_{X/Y/Z}$	$X/Y/Z$ component of $\mathbf{A}^B$
$\mathcal{A}$	List
$\mathcal{A}(i)$	ith element in $\mathcal{A}$

Letters

$\mathbf{a}$	Acceleration vector
$\mathbf{A}$	System Matrix
$\mathbf{B}$	Input Matrix
D	Down
E	East
$\mathbf{E}$	Attitude
f	Focal length, update frequency
$\mathbf{g}$	Gravity vector, gradient vector
g	Gravitational acceleration
$\mathbf{h}$	Optimisation step direction

$\mathbf{H}$	Output matrix, Hessian
$\mathcal{J}$	Cost function
$\mathbf{J}$	Jacobian matrix
L	Euclidian Distance
$\mathbf{L}$	Kalman gain
M	Mass
N	North
$\mathbf{O}$	Origin
$\mathbf{p}$	2-D projected position vector
$p,\ q,\ r$	Roll, pitch and yaw rates
$pp_{x/y}$	Principle point in x/y direction
$\mathbf{P}$	3-D position vector, Propagated state covariance
$\mathbf{Q}$	Process noise covariance matrix
$\mathbf{R}$	General rotation, Measurement noise covariance matrix
$\mathbf{T}$	General translation
$\mathbf{v}$	3-D vector to 2-D image plane point
$\mathbf{v}$	Measurement noise
$\mathbf{V}$	Velocity vector
$\mathbf{w}$	Process noise
$\mathbf{x}$	State vector
$\mathbf{y}$	Measurement
w_{px}	Pixel width

Greek Symbols

ϕ	Roll angle
θ	Pitch angle, centroid separation angle
ψ	Yaw angle, absolute centroid angle
σ	Standard deviation
α	Optimisation step size
λ	Component, Ageing factor
τ	Time constant
Δ	Determinant
$\mathbf{\Phi}$	Discrete System Matrix
$\mathbf{\Gamma}$	Discrete Input Matrix
ϵ	Threshold

Subscripts

pv	Position and Velocity

att	Attitude
GPS	Related to GPS measurements
Cam	Related to camera measurements
ref	Reference
$hover$	Related to hover condition
WP	Waypoint
T	Target
$\perp$	Perpendicular
d	Distorted
u	Undistorted
COD	Center of Distortion

Superscript

n	Waypoint navigation related
l	Landing related

Abbreviations and Acronyms

ATOL	Autonomous Take-off and Landing
bps	Bytes per Second
CAN	Controller Area Network
DAC	Digital to Analog Converter
DGPS	Differential Global Positioning System
DOF	Degree of Freedom
EKF	Extended Kalman Filter
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
HIL	Hardware-in-the-Loop
IP	Intellectual Property
IR	Infrared
KF	Kalman Filter
MIPS	Million Instructions per Second
NED	North East Down
OBC	Onboard Computer
OTS	Off-the-Shelf
RMS	Root Mean Square
RUAV	Rotary Wing Unmanned Aerial Vehicle
SVD	Singular Value Decomposition
TPP	Tilt Path Plane

TQFP	Thin Quad Flat Pack
USB	Universal Serial Bus
VPAM	Vision-based Position and Attitude Measurement

Chapter 1

Autonomous Landing

1.1 Introduction

1.1.1 The Relevance of Autonomous Take-off and Landing

One of the most commonly cited advantages of a helicopter is its VTOL (vertical take-off and landing) ability [5] which is especially important in situations where the space required for conventional aircraft landings is not available. Another unique ability of VTOL aircraft is their ability to hover. As a result, such vehicles are ideal for surveillance and search and rescue missions. Autonomous vehicles have the advantage of performing tasks where human intervention is impossible, dangerous, expensive or monotonous [5]. As the accuracy and speed of sensing technology and small computational platforms increases, their ability to perform high precision tasks in challenging and uncertain scenerios improves [75]. The aim of this research is to develop the necessary low cost autopilot systems that will, in future, allow a rotary wing unmanned aerial vehicle (RUAV) to behave autonomously for the entire flight envelope.

1.1.2 The State of RUAV ATOL Research

Within the scope of unmanned aerial vehicles (UAVs), autonomous rotary-wing aircraft have matured into a major research topic [75]. This is evident by the number of renowened academic institutions having dedicated research groups: University of California at Berkeley [76, 86, 87, 78, 80, 79], University of Southern California [72, 71, 70, 82, 83], Boston University [42] and Carnegie Mellon [6]. These research groups have done work on basic RUAV flight, waypoint navigation, acrobatic flight, strategic mission execution, ATOL, rendevouz etc. Autonomous RUAV landing has received considerable attention since landing is a crucial component of any RUAV mission [82]. This section will outline the current state of RUAV autonomous landing research.

Autonomous landings can be classified according to the landing target (Figure 1.1) and the choice of sensors used to facilitate the landing (Figure 1.2).

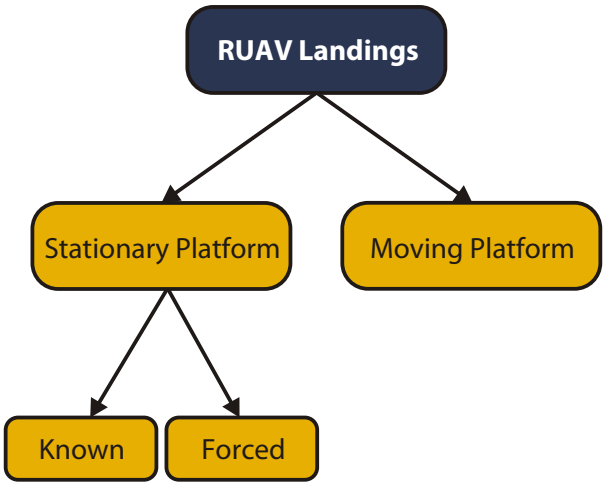


Figure 1.1: Classification of RUAV landing methods

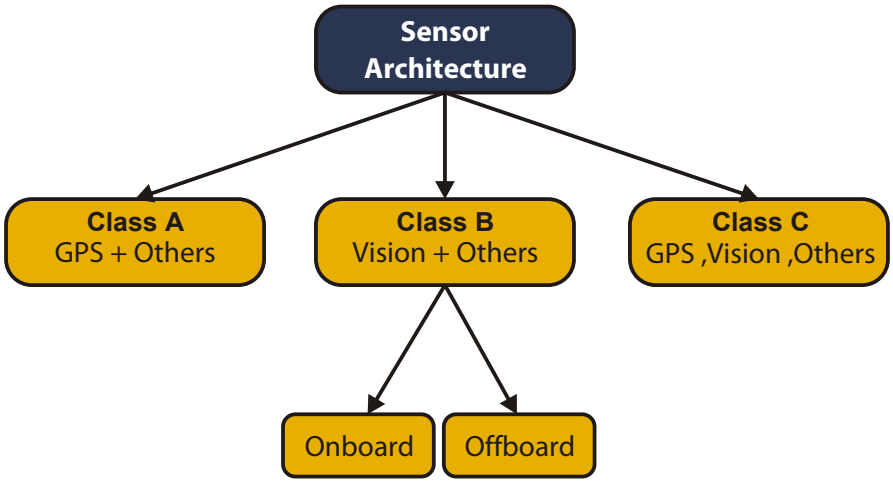


Figure 1.2: Different approaches to sensor arrangements for RUAV ATOL

Known (Man-made) Stationary Landing Target

Autonomous landing on a known stationary target is outlined in [82, 88, 78, 65, 83, 32]. This is the most basic form of autonomous landing from both a measurement and estimation, and control perspective. Measurement and estimation is simplified since the location and appearance of the target are known, which reduces processing requirements. Control is also simplified since the target does not move during the landing sequence.

Position accuracy with differential global positioning system (DGPS) can be as accurate as 2 cm [60], but the relative attitude between the vehicle and the landing target cannot be determined without information describing the landing target's attitude. There are at least two methods that can be used to determine the vehicle attitude. The first method employs multiple DGPS receivers placed at different locations on the vehicle. The phase information from the GPS packets can be used to determine attitude. The alternative is to use measurements from the inertial measurement unit (IMU) only and could only be attempted practically if very low drift gyroscopes are used. If the landing target's attitude

is fixed, the state estimator must be aware of this attitude. If the landing target is moving and has a variable attitude, measurements from the platform must be transferred to the estimator to determine the relative attitude.

Architectures using only GPS to perform autonomous landing are classified as Class A (in this thesis) in Figure 1.2 and require high accuracy (and as a result expensive) avionics to be successfully implemented. It also has no feedback about the landing target which will not allow an easy transition to autonomous landing on a moving platform. As a result, this architecture will not be considered for this project.

During the landing procedure, the relative position and attitude between the vehicle and the landing target are required. Autonomous flight has historically utilised a wide array of sensors to determine the position and attitude of the vehicle [65]. For absolute position, offboard data is often required (GPS or radar for instance). This is in contrast to the natural behaviour of animals and humans: birds of prey, for instance, do not require ground personnel to send the coordinates of their meal before they attack. Since a human pilot can visually acquire a specified destination and fly to it, it stands to reason that the same could be achieved using only computer vision [65, 88]. Such systems, classified as Class B, only employ visual methods to provide relative position and attitude feedback. Advantages of such systems are robustness (since they are not dependent on communication to the vehicle – thus insusceptible to jamming and interference) and accuracy [88]. Examples of such systems are described in [88, 65, 78].

A vision based navigation system was developed in [88] with a sufficient operating range to allow low latency robust pose estimation of a specially designed landing target (Figure 1.3). This reference pattern was chosen to allow the system to have fast recognition, accurate pose estimation for varying ranges, small size and minimal symmetry. The high contrast design simplifies the detection while providing accurate image features. After detection and extraction, the distortion of the circles on the target due to the relative attitude is used to determine the vehicle's relative pose. To extend the operating range the camera is mounted on a pan/tilt platform to obtain visual feedback from the landing target irrespective of the vehicle's attitude. A standard Kalman filter was modified to make use of the visual measurements when they were available to improve the accuracy of position, velocity and attitude estimates. A finite state machine approach was used as an outer loop to create inner loop references, guiding the helicopter to touch down .

A number of tests in different wind conditions and from different starting positions were performed with an average translation touchdown error of 42.6 cm. The maximum error was 54 cm while the minimum error was 22 cm. The largest error was obtained during the highest wind speed of 30 km/h.

A vision only (Class B) approach to provide accurate state estimates that will allow an RUAV to land autonomously is described in [65]. The image processing was simplified by using red beacons to mark the landing zone and separating the red layer at the start of processing. Since bright red is relatively rare in nature, the resulting image has very little

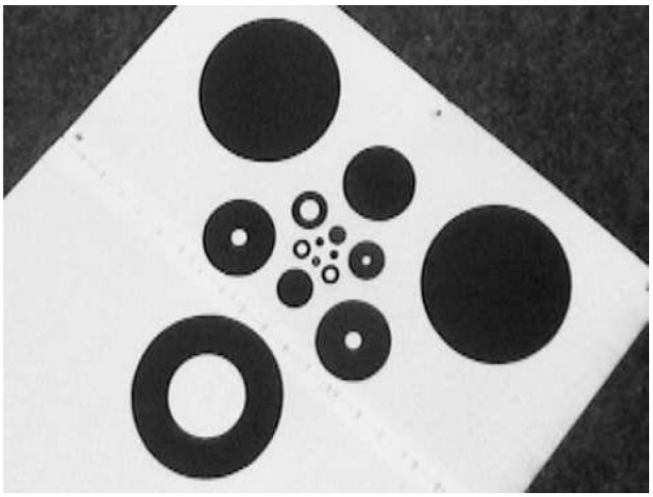


Figure 1.3: Custom landing target used for navigation and autonomous landing [88]

Mode	Horizontal control	Yaw control	Altitude control	Logical condition for Mode Transition
READY				
AIM	hold	$\psi \rightarrow \psi_{AIM}$	hold	$\lvert\psi-\psi_{AIM}\rvert < 5°$
APPROACH	linearly moving to P_D @ V_{HOR}<0.5 m/s	hold	Descending to PD $h \rightarrow h_{D1}$ (5 m) @ VZ = -50 cm/s	$\lvert P-P_{D1}\rvert < 2$ m $\lvert h-h_{D1}\rvert < 0.4$ m $\lvert V_Z\rvert < 10$ cm/s $\lvert V_{HOR}\rvert < 0.3$ m $\lvert\psi-\psi_{AIM}\rvert < 3°$
ALIGN	hold	$\psi \rightarrow \psi_{TOUCHDOWN}$	Hold	$(\psi-\psi_{TOUCHDOWN}) < 3°$
DESCEND	hold	hold	$h \rightarrow h_{D2}$ (1 m) @ VZ = -20 cm/s	$\lvert P-P_{D2}\rvert < 0.25$ m $\lvert h-h_{D2}\rvert < 0.1$ m $\lvert V_{HOR}\rvert < 0.3$ m $\lvert V_Z\rvert < 0.1$ m/s vision data VALID
TOUCH DOWN	hold	hold	VZ = -20 cm/s	h < 0.1 m
SHUT OFF	hold	hold	Throttle back descending ramp	

Figure 1.4: Algorithmic state approach for landing guidance [88]

noise which results in distinct markers after thresholding. Pose estimation is performed by a Kalman filter, resulting in a relatively high computational demand. As a result, it was implemented on a dedicated onboard 833 MHz Pentium 4 computer. Outer loop guidance controllers use the accurate estimates to guide the RUAV to the landing target in a similar fashion to a fixed wing aircraft. The system functionality was only confirmed through simulation.

A high frequency vision system to assist in the autonomous landing of a RUAV is outlined in [78]. The vision system is divided into two stages: image processing and state estimation. Image processing was identified as a potentional bottleneck and the landing target was designed to simplify detection, simplify segmentation from the background and provide distinctive features points. The target is shown in Figure 1.5. The custom target allows the algorithms executing in real time using off the shelf (OTS) hardware.

A classic image processing approach of thresholding, segmentation, corner detection and feature labelling is employed. The resulting feature points are used in an iterative, optimisation based, pose estimation algorithm to determine the relative helicopter position and attitude states. To determine the measurement accuracy, the visual measurements were compared with 2 cm accurate DGPS measurements. Position accuracy was 5 cm in each axis of translation and attitude accuracy was within 5°. The system, running on a dedicated 233 MHz processor, provided estimates at 30 Hz, with 90% of the processing time spent on low level image processing (segmentation/thresholding) despite the measures taken with the custom landing target.

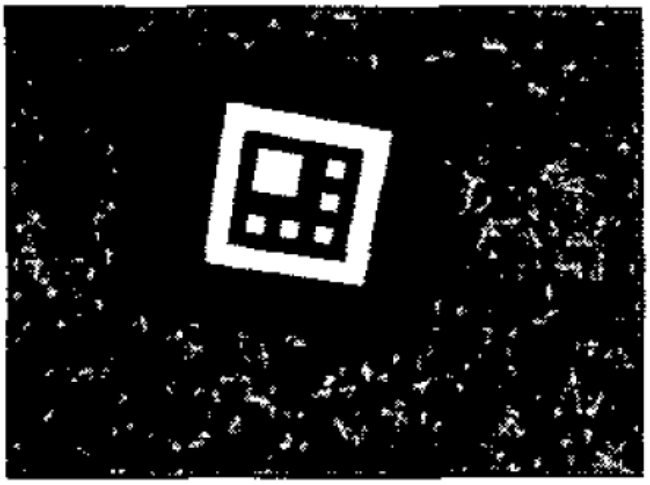


Figure 1.5: Custom landing target to simplify processing [78]

A vision system for pose estimation of a Class B quadrator RUAV is described in [4]. The method consists of a pair of ground based and onboard cameras to estimate the full six degree of freedom (6-DOF) motion of the helicopter. The aim is to allow small UAVs to perform autonomously in indoor environments where GPS does not function. Limited payload capability of the vehicle forced the vision processing to be done offboard. A tilt/pan camera on the surface processed five different-coloured markers placed on the quadrotor, while the onboard camera tracked a single marker on the ground. The results of the pose estimation were transferred to the vehicle using a tether. The dual camera arrangement is unique since the subject for both cameras is not the same. Both the translational position error (1.27 cm) and angular error (1.22 °) are smaller than the errors obtained with a conventional monocular view system (3.08 cm and 3.05° for position and angle respectively). Basic control to hover and command the quadrotor has been achieved, with autonomous landing identified as a future development using this system. Like differential GPS, however, it is dependent on a communication link to perform landings, which reduces the robustness of the system.

A vision-only approach (Class B) with known markers imposes a severe constraint on the operating range of the RUAV. A more practical approach is a combination of GPS and a vision system (Class C architecture in Figure 1.2). A very basic version of such

a system is described in [82] and [83]. Here GPS is used whenever visual measurements are not available (due to range or occlusion). A high intensity target (Figure 1.6) is used to simplify the target detection during image processing and invariant moments are used to perform the extraction and detection. When visual measurements are available, the centroid positions in the image frame are simply converted to a relative translational position and yaw angle using the DGPS altitude measurement as reference. The high level guidance controller is similar to the one described in [88] where the vehicle is hovered above the target before a constant descent is initiated. The specifications of the processor are not published, but the system achieved a 10 Hz rate with low level tasks (thresholding and segmentation) taking more than half of the processing time. Fourteen autonomous landings were achieved with a 6° mean heading error and a 42 cm mean position error.

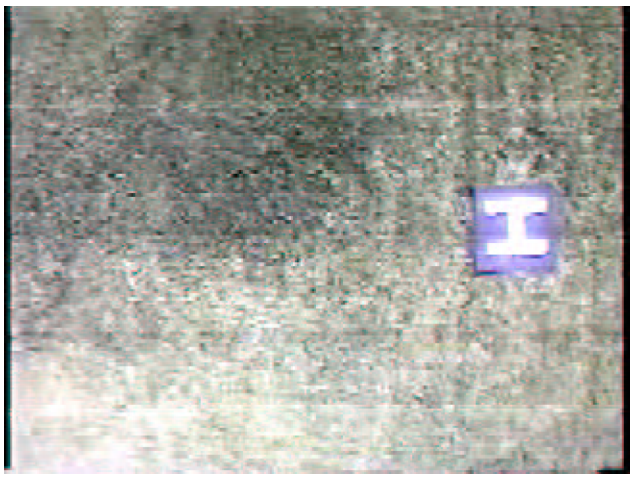


Figure 1.6: Frame from onboard camera showing the landing target [83]

Unknown Stationary Landing Target (Forced/Safe Landing)

There are many situations where a RUAV landing would not be able to take place on a known man-made target, such as perch-and-stare reconnaissance missions [86, 11], emergency landings [86, 11, 53] and hazard avoidance [39, 54]. Unless a detailed map of the surrounding terrain with possible landing locations is known, GPS would not be sufficient to perform such manoeuvres. When faced with the situation of landing in hazardous terrain, a human pilot would look for a safe location and make a decision based on what he/she sees. It stands to reason then, that it would be possible to determine safe location using computer vision techniques. This premise has been investigated and was confirmed in [98, 29, 86, 39, 11, 54].

A stereo-vision system to determine the range of the environment through sum of absolute difference correlation between the left and right images is described in [98]. It is assumed that the terrain is flat and the altitude is unknown. The inner loop controllers were designed using the linear quadratic control strategy, while the references were determined through a layered outer loop controller. The system's functionality was confirmed through simulations with centimetre level accuracy.

A decision strategy for safe landing using computer vision is presented in [29]. The strategy is not constrained to any particular environmental conditions. A safe landing target is defined as any area on the ground that is large enough for the RUAV to land within and free of natural or artificial obstacles. A third condition, not covered by this research project, is to require the target to be a piece of ground where the curvature is zero and where the local ground plane is horizontal. (This condition is included in the methods described in [86] and [39].) The area is detected through a process of contrast thresholding and finding a circular area of sufficient size below the threshold. Finding the area in the image is done through a combination of optical flow and motion tracking. The algorithm executed at over 30 frames per second on a dedicated OBC. Decision accuracy was good, with only a 2% false positive rate. Actual landings were not performed using the selected target.

A Class C system, where a monocular vision system is used to determine the coordinates of a safe landing site and the GPS is used for feedback during the navigation and landing procedures, is outlined in [39]. Successive images taken during flight are used as an approximation to a stereo-vision setup to determine depth and roughness information. Based on the partial maps generated during flight, a safe landing target is chosen. Site detection is done in 720 ms on a dedicated 400 MHz OBC. The average position error obtained for landings during three test flights was 1.1 m, with the weather conditions not being reported. A similar approach was described in [86], and succesful landings were achieved, but the accuracy of the results was not reported.

Known (Man-made) Moving Landing Target

For shipboard operations, landing on a moving platform is a critical manoeuvre [24]. Landing on a platform experiencing large attitude deviations poses more involved estimation and control problems than when compared to a stationary platform. The motion of the deck now plays a very important role in the relative position and attitude estimation as well as the trajectory planning and decision making during the landing process. Relative state estimation generally involves the tracking of the platform using visual methods [30, 97], radar guidance [12] or ship-based navigation beacons (pseudolites) [57]. Radar and ship-based beacons rely on a low bandwidth datalink which introduces a single point of failure into the system as well as making the system susceptible to jamming [30].

The landing procedure on moving platforms are usually divided into two distinct phases [24, 30]: initial attenuation of the relative position error and platform tracking for a safe touchdown. The first phase could be performed by GPS (if the position of the target is known) or vision methods (if the appearance of the target is known) to get the RUAV to a position above the target. To perform accurate tracking during the second phase, accurate relative attitude and altitude data is required [30, 97, 61, 71]. Once accurate measurements of platform movement are available the control system needs to facilitate the landing process [45, 24, 61, 71].

A precision guidance system that incorporates an accurate estimate of the ship's motion

and measurements of the instantaneous orientation and location of the landing deck is proposed in [30]. The guidance system has two components. The first makes use of a monocular vision system with a single 650 nm light source on the target to align the RUAV with the center of the RUAV. A single light source was chosen to maximise available field of view and still provide the two dimensional position error information. The wavelength was chosen due to less interference by water and mist than at near-infrared wavelengths. The second component generates an array of 3-D coordinates through a laser rangefinder and spinning mirror that projects an oval onto the target. Through a process of least squares estimation, the relative distance and orientation of the target with respect to the helicopter can be determined. The reported attitude accuracy is within 1° and altitude accuracy in the centimetre range. Testing of the prediction algorithm and landing controllers had not been performed at the time this thesis was documented. The platform during estimator testing is shown in Figure 1.7.

Figure 1.7: R-Max RUAV hovering above the moving deck simulator [30]

A real-time stereo vision based pose and range estimation system is presented in [97]. Making use of a known target marker, stereo cameras are used to determine the relative pose between the helicopter and the landing platform. Each iteration of the algorithm executed in 175 ms on a 1.4 GHz Pentium desktop computer with angular RMS errors in the order of 0.8° and translational position RMS errors below 5 cm. The altitude RMS error was 30 cm at 8 m above the target and 3 cm at 2 m above the target. Future work recommended in this project includes the integration with a navigation system and controllers to faciliate the landing of an RUAV on a moving platform.

The use of a tether to determine the relative motion between the helicopter and the

landing target is described in [61]. The simulated landing accuracy is within 40 cm.

A complete RUAV landing system for moving platforms is presented in [70]. The problem is decomposed into four stages: detection of a custom landing target using vision, tracking of the target using a Kalman filter, motion planning to determine a trajectory for the landing and control to regulate the RUAV to the planned trajectory. For this iteration, platform motion was constrained to lateral displacement only and simulation results confirmed successful trajectory tracking and landing.

Other Vision-based RUAV Applications

In the preceding three sections a variety of systems and methods developed to perform autonomous RUAV landings were discussed. Vision based systems are preferred to GPS systems due to a more detailed level of environmental feedback, robustness and cost. Vision systems, however, can be used for much more than just measurements and surveying during the landing phases. Inspection [6, 18], aerial mapping [6], cinematography [6], navigation using urban features [54] and navigation using landmarks [43, 50] are also potential applications.

1.1.3 Research At Stellenbosch University

The first autonomous RUAV flight achieved at Stellenbosch University is outlined in [10]. A small electrically powered helicopter with very limited payload capabilities was used. As a result, estimation and control were done offboard and with the actuator commands transmitted back to the vehicle via a radio link. The latency in this architecture, combined with the low quality sensors, limited battery capacity and changing dynamics (due to fluctuating blade speed led) to a very limited system. Despite this, the validity of low cost RUAV flight using successive loop closure principles was confirmed and paved the way for more advanced RUAV research.

At the start of the restructured RUAV research project in 2004, a size 70 glow-powered helicopter with significant payload capability (2 kg) was acquired that allowed the onboard execution of state estimation and control algorithms [10]. The development of the avionics, preparation of the platform and first iteration of the non-linear helicopter model is outlined in [31].

In 2006, two concurrent RUAV projects were initiated with the aim of accelerating the progress toward autonomous flight [69]. Different project outcomes were defined, but platform work was shared between the two projects – reducing time spent by each individual on non-academic work needed to achieve autonomous RUAV flight. Research to achieve hover-based autonomous flight is outlined in [69]. This involved the further development and analysis of the non-linear helicopter model, the development of a decoupled linear model for control system development and the development of a successive loop closure control system. Groundstation software was developed to provide feedback and manage the autopilot during flight. The entire system was verified through a hardware-in-the-loop

(HIL) system [37] and a number of flight tests. These tests marked an important milestone in RUAV research at Stellenbosch University. Full-state feedback strategies (such as the linear quadratic regulator) were investigated and developed in [91]. This allowed the helicopter to perform significantly more advanced manoeuvres such as barrel rolls, loops and stall turns. Due to the high computational complexity and high risks during flight testing, these methods were confirmed in HIL only.

In 2007, a series of research topics with the common long term goal of autonomous take-off and landing (ATOL) on a ship's deck was defined for fixed- and rotary-wing aircraft. Two projects in this series were commenced in 2007: the development of a three degree of freedom platform simulating a boat's movement [81] and the precision landing of a fixed wing UAV on a stationary platform [92].

The aim of the project was to land a fixed wing aircraft on a stationary 10 m $\times$ 10m platform, within a radius of 1 m [92]. The required accuracy could not be achieved using the low-cost IMU and GPS module used in previous projects and alternatives were investigated. A vision based approach was chosen with infrared lamps and an infrared sensitive camera, reducing the complexity and runtime of the image processing. The standard state estimator was modified to use the vision measurements to improve the position accuracy to 0.1 m and the attitude accuracy to within a degree. Acceleration based controllers were used in conjuction with the necessary outer loop controllers to guide the aircraft onto the correct glide slope. Practical testing confirmed the functionality of the system and signalled the start for more intelligent vision-based developments at Stellenbosch University.

The project outlined in this thesis can be seen as an extension of the work described in [31] and [69] with influence from the developments outlined in [92]. The main goal of this project was to advance the autonomous capability of the RUAV by implementing high level functionality such as landing and waypoint navigation. The main goal was subdivided in a divide and conquer approach:

1. Literature study to investigate the state of RUAV ATOL research worldwide.
2. Redevelopment of the camera hardware and algorithms outlined in [92] to make it more modular, efficient and portable. Due to a large number of recommendations regarding the vision system in [92] and the potential for significant improvements, this would be the main focus of this project.
3. Development of the necessary outer loop controllers to perform autonomous landing on a stationary platform within a circular region with 1 m radius.
4. Development of the necessary guidance controllers to perform waypoint navigation.
5. Verification through HIL simulation and test flights.

At the conclusion of this project, four related projects were underway at Stellenbosch University to further accelerate the achievement of the overall project goal: landing on, and taking off from a moving platform for fixed wing and rotary wing aircraft.

During the first part of this project a redundant inertial measurement unit capable of providing measurements despite sensor failures was designed and implemented. Fault detection and isolation was performed using a novel, vector-based approach to detect up to two sensor failures and isolate a single failure. The functionality of the unit was confirmed using simulation and practical testing. The development is outlined in a paper published and presented at the European Space and Air Conference in October 2009 [17]. Since the main focus of the project shifted to vision-based position and attitude measurement applied to autonomous helicopter landing, the IMU development is not discussed in this thesis.

1.2 Landing Procedure

Landings on a stationary platform can be divided into a number of categories, based on the angle of descent to the landing target [2](Figure 1.8). For descent angles smaller than 90° the approach is called a sloped surface approach (or no-hover landing) and is further classified according to the angle of descent as normal, steep or shallow. A manoeuvre starting from a hover with a 90 degree descent angle is known as a vertical surface approach. The choice of approach depends on the prevailing conditions which may include obstacles, size and surface of the landing target, air density, wind direction and speed, and aircraft weight.

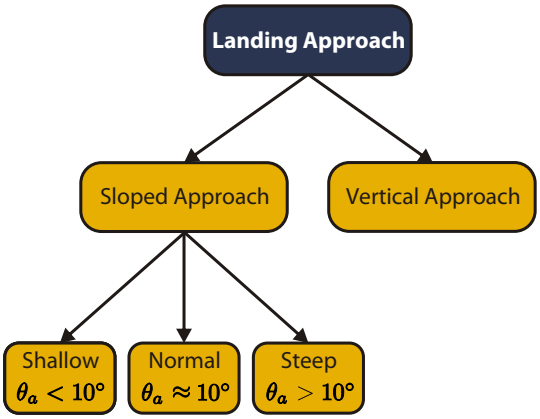


Figure 1.8: Landing classification according to approach.

1.2.1 Sloped (No-hover) Approach

Sloped approaches are chosen when hovering over the landing target is not possible due to surface conditions eg.loose snow or dust and/or power constraints (due to gross weight and or air density). The choice of normal approach (10° descent angle), steep approach or shallow approach depends largely on obstacles and available power. The less power

available to hover, the shallower the approach will be. To compensate for the lack of power, a shallow approach makes use of translational lift until touch down is achieved. Since the helicopter will be sliding/rolling to a stop, the landing area must be smooth and long enough to accomplish this task. A steep approach is used when there are obstacles in the way of a normal approach path. It can also be used to avoid areas of turbulence.

During any sloped/no-hover approach (Figure 1.9) airspeed is slowly bled off by executing a flare manoeuvre [93]. A flare is initiated by an aft movement of cyclic, tilting the tilt path plane (TPP) rearwards (B-D in Figure 1.9). The horizontal component of the total rotor thrust is in the opposite direction to the motion, which reduces airspeed. By controlling the vertical component of the total rotor thrust through collective, the rate of descent can be controlled. Longitudinal trim is maintained by using the tail rotor [2]. For wheeled helicopters, touchdown with non-zero forward speed is allowed, given a smooth and long enough landing target. For conventional helicopters, however, touchdown should occur with skids level, zero groundspeed and a descent rate approaching zero (E in Figure 1.9).

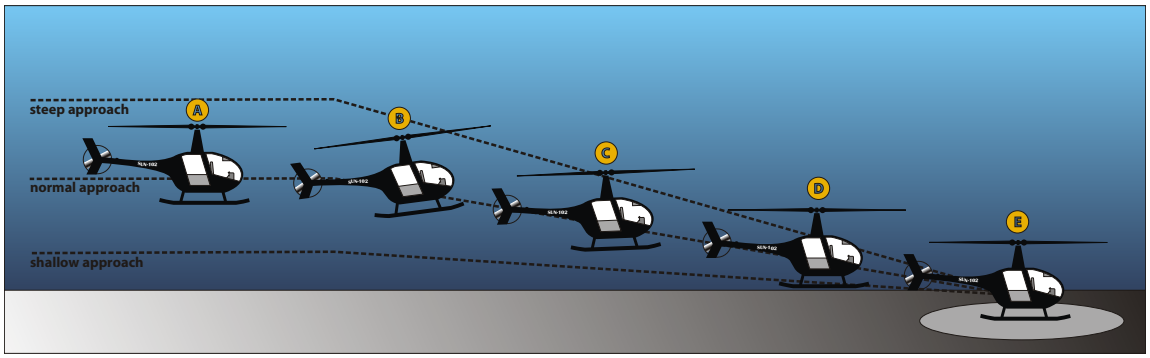


Figure 1.9: Conventional landing approaches. A normal approach (10°) is shown with a shallow approach at 5° and a steep approach around 15°.

1.2.2 Vertical Approach

Vertical approaches (Figure 1.10) are suitable when hovering over the landing target is possible and the target surroundings prevent a normal approach to be used. Vertical approaches involve performing a flare manoeuvre (B in Figure 1.10) to reduce airspeed without losing altitude and settling directly above the landing target in a hover (C in Figure 1.10). Altitude during the flare is controlled by the vertical component of the total rotor thrust through collective. The descent is performed by reducing the collective until the desired rate has been obtained. Cyclic and tailrotor adjustments are used throughout the descent to keep the vehicle lined up with the landing target until touchdown has been achieved (D in Figure 1.10).

1.3 Proposed System Overview

This section will give a brief overview of the system outlined in the remainder of this document. Decisions that had a major effect on the system design are also explained and

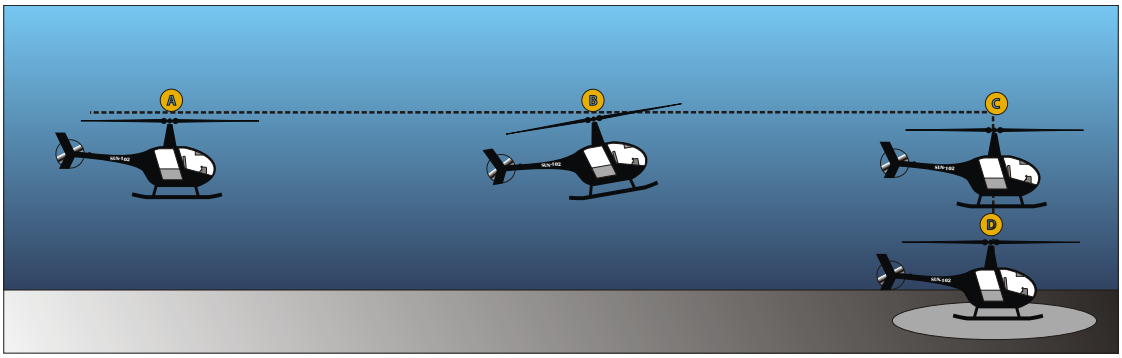


Figure 1.10: Vertical landing approach

justified. The complete system is shown in Figure 1.11.

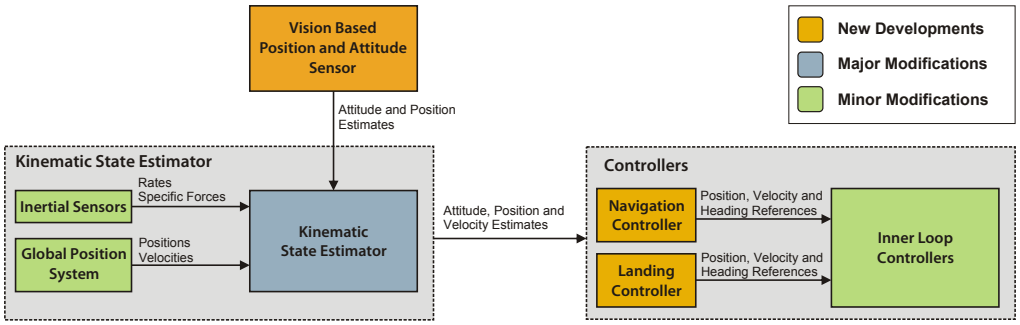


Figure 1.11: Autonomous RUAV landing overview

A low-cost GPS receiver does not provide measurements with the required accuracy for autonomous landings [92]. Vision-based systems and differential GPS were identified as alternatives in the literature study (Section 1.1.2).

From the comparison shown in Table 1.1, it was decided to make use of a vision based system to provide high accuracy measurements during the landing phase, but maintain low cost GPS for measurement updates when the landing target is outside the field of view. The principle disadvantage of this approach is the computational complexity associated with computer vision algorithms. This complexity is mainly due to many computer vision algorithms repeatedly iterating the image. This can, however, be limited by reducing the image elements without discarding useful information.

It was also decided, from a risk aversion point of view, to focus on vertical landings. With no-hover landings, forward velocity needs to be zero at touch down. In addition, the descent rate must approach zero at the same time to avoid very hard landings. If these conditions are not met at touchdown, there is a high risk of roll-over and resulting vehicle damage. With vertical landings, the risk of roll-over is significantly lower. Once the system has been proven to work with vertical landings, it can be extended to perform no-hover landings.

Table 1.1: Comparison between DGPS and Vision based systems

DGPS	**Advantages** ▶ Accurate position and velocity information ▶ Valid over entire flight envelope ▶ No additional processing required/lightweight ▶ No (direct) attitude information ▶ No landing target feedback
	Disadvantages ▶ Expensive ▶ Data link, single point of failure ▶ Data link, open to jamming and interception
Vision	**Advantages** ▶ Accurate position and attitude information ▶ No data link required (for onboard processing) ▶ Landing target feedback ▶ Flexible technology (mapping, forced landing)
	Disadvantages ▶ Generally only valid for a section of the flight envelope ▶ Computationally expensive, heavy payload

Once accurate measurements are available, they need to be incorporated into a state estimator where high frequency inertial measurements are combined with low frequency updates. These state estimates can then be used to stabilise and command the helicopter to certain positions, velocities and attitudes.

Landing would be overseen by a landing controller integrated with the existing controllers using a hierarchical approach. The landing controller is responsible for generating the correct inner loop references for every step in the landing process as outlined in Section 1.2.2.

1.4 Document Overview

This document is structured as follows:

Chapter 2 The development of the computer vision algorithms to provide accurate state information.

Chapter 3 The development of a visually augmented state estimator to fuse visual measurements with measurements from inertial sensors and GPS.

Chapter 4 The development of navigation and landing algorithms to perform high level guidance.

Chapter 5 A description of the design and implementation of the camera hardware.

Chapter 6 A description of the simulation environments used during the project to confirm functionality. Results of these simulations are shown and analysed.

Chapter 7 A discussion of the practical aspects of the camera hardware and a description of the test configurations. Practical results are presented and analysed.

Chapter 8 This thesis concludes with a number of deductions that can be drawn regarding the development, implementation and testing of a vision-based position and attitude measurement system for the purpose of autonomous rotorcraft landing and also provides a number of recommendations for future research.

1.5 Summary

In this chapter different landing procedures were discussed and comparable research projects were evaluated in a literature study. The placement of this project within the Stellenbosch University rotorcraft project was also explained. A high level overview of the system was developed, keeping the aim of low-cost, low-weight systems in mind.

Chapter 2

Vision Algorithm Development

2.1 Introduction

Low-level vision algorithms tend to be computationally expensive which increases the processor requirements. More powerful processors are more expensive and demand more power. This leads to a heavier avionics system, which reduces the payload capabilities of the vehicle. Clearly then, simplified algorithms are necessary to create a low cost/low weight system without dramatically sacrificing functionality or accuracy. The first iteration of a system with the potentional of being implemented on a low power hardware is described in [92]. This system, however, was still implemented on a reasonably powerful processor (300 Mhz Celeron). This chapter outlines the simplified algorithms and methods required by the vision system to convert images of the landing target into high accuracy relative position and attitude measurements.

Low-level image processing algorithms are computationally expensive because they generally have to perform calculations for every pixel (or regions of pixels). If the image can be simplified, however, without sacrificing information that is necessary for operation, the low-level algorithms can be simplified. It was shown by [92] that the image can be simplified by enforcing the following two conditions:

1. The landing target must be completely described through a small number of discrete feature points.
2. Images must contain mainly valid feature points with very few other environmental features.

The first condition is easily met by using a geometric shape as the landing target. The target can then be fully described by an ordered listing of the vertices (Figure 2.1).

The examples shown in Figure 2.1 are both symmetric and the relative yaw angle cannot be determined from an image of such a target. This can easily be solved by adding an additional feature point to the target (Figure 2.2).

Figure 2.1: Geometric landing targets

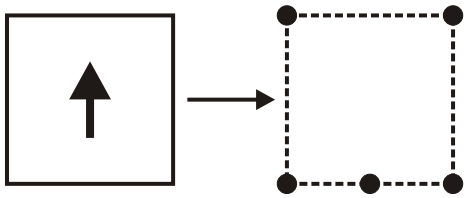
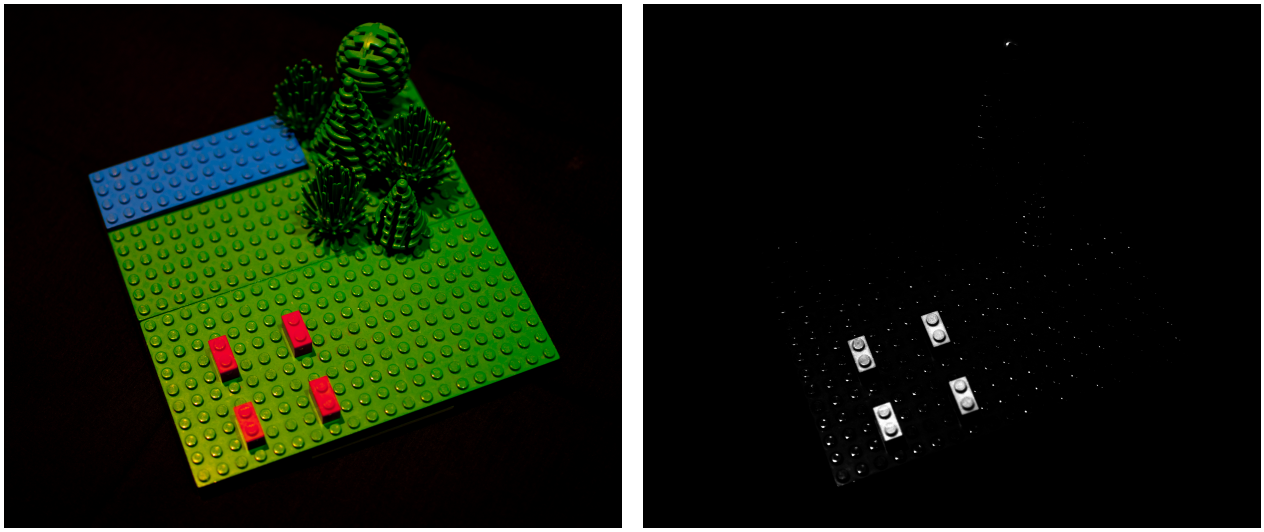


Figure 2.2: Geometric target with an additional feature point to remove rotational symmetry

Details of how the second condition is satisfied with very little (or no) increase to the computational complexity is discussed in Chapter 5, but a brief overview is given here. The environment can be surpressed in images if the landing target is very different from the environment in some aspect (such as colour). In [65] this idea was employed by using red balls as target markers in a predominatly green environment. By extracting the red layer from a colour image, and thresholding the resulting image, the projections of the landing markers could be detected easily. A replica of this system is demonstrated in Figure 2.3. Obviously, there are natural objects that have red content (such as rocks, sand, reflections from water, etc) which will introduce some phantom centroids. Reflections from the fill light is visible in the bottom corner of Figure 2.3 for instance. In most cases these can be rejected by the algorithms presented in this chapter. The method employed in this thesis makes use of infrared lights as markers that can be easily detected by an infrared sensitive camera. The specific wavelength of light usedis very rare in nature - further improving robustness. A more detailed explanation on the design of this approach is done in Chapter 5, but for now it is sufficient to assume that both conditions are satisfied.

The basic flow of a conventional vision-based position and attitude measurement (VPAM) system [78, 82, 88] is shown in Figure 2.4. Low level image processing (which operates on the full image) has been identified as the bottleneck in VPAM systems [78].

The basic flow of the proposed vision system is shown in Figure 2.5 where the processing is done on a simplified image from the start due to the two constraints imposed on the landing target. The output from the camera is a list of all plausible high intensity regions in the image. These are determined in a pipeline fashion while the image is transferred from the image sensor (Chapter 5). The next step in the process is correspondence matching, where it is determined which projection in the image best corresponds to each actual marker. Through an iterative approach the best match of the landing target is



(a) RGB image (b) Thresholded red layer

Figure 2.3: Colour differentiated target

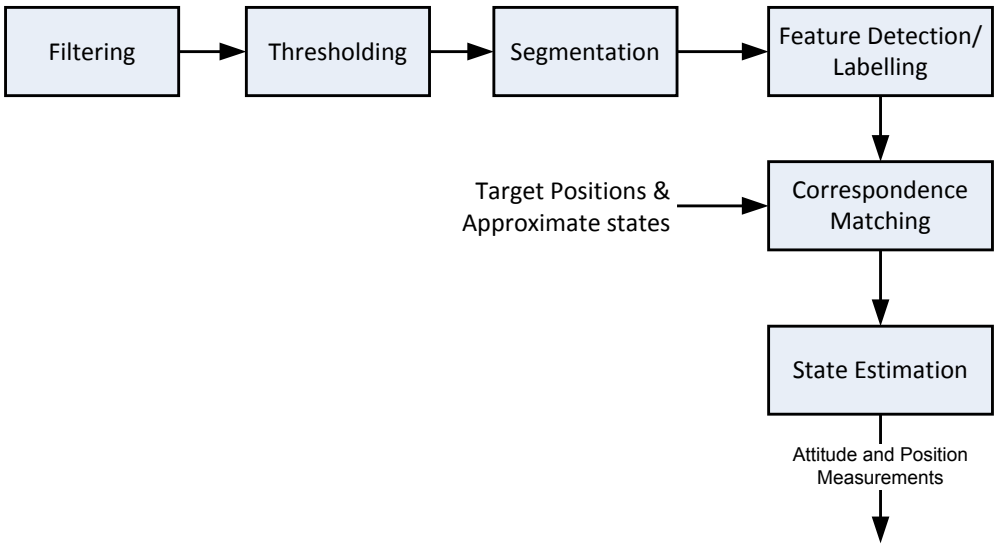


Figure 2.4: Overview of a conventional VPAM system

determined and possible noise centroids are rejected (Section 2.5). Once the appropriate centroids have been identified, they can be used in conjunction with *a priori* geometric knowledge of the landing target to determine the relative position and attitude (Section 2.6).

2.2 Reference Frames

A variety of reference frames are used to model the camera system and perform the position and attitude estimation from the images:

World Reference Frame/NED Frame An inertial frame is non-rotating and not accelerating, and it is generally used as an absolute reference. Traditionally, in com-

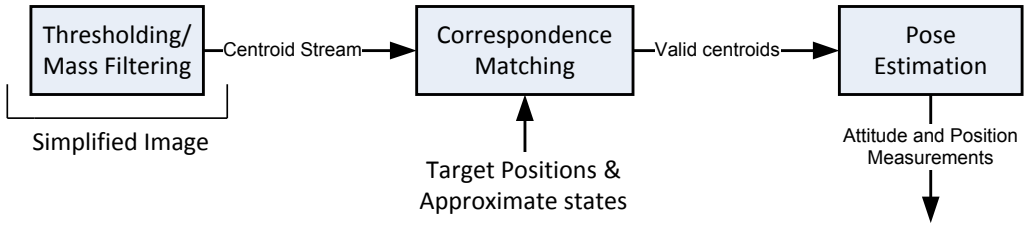


Figure 2.5: Overview of the simplified VPAM system

puter vision applications, this frame is called the world reference frame [16, 77]. In UAV applications, however, the NED plane (also called the local tangential plane) is considered an inertial frame [37] with the North and East axes pointing in the appropiate directions and the Down axis pointing towards the center of the earth. For this reason, the NED frame will be used as the world frame in the computer vision algorithms

Camera Reference Frame The camera reference frame is fixed to the camera with the z-axis pointing forward along the camera's optical axis. The y-axis points upwards while the x-axis completes the right handed triad. Coordinates in this frame have the same units as the NED frame. The origin of this frame is placed at the aperture of the camera and is referred to as the optical center.

Real Image Reference Frame The real image reference frame is a two-dimensional cartesian frame attached to the image plane onto which the image is projected. The origin of the frame is at the intersection of the optical axis with the image plane. Coordinates in this frame have the same units as the NED frame.

Pixel Coordinate Reference Frame Since the algorithms will be implemented using a digital image sensor, the pixel coordinate reference frame is introduced. This two-dimensional frame is attached to a quantised version of the image frame and the coordinates are discrete row and column positions.

2.3 Basic Camera Model

The simplest camera model is the pinhole[1] model and it describes the mathematical relationship between the coordinates of a point in a three dimensional scene and its projection onto the image plane of an ideal pinhole camera. In such a camera, the aperture is described as a point and no lenses are used to focus light from an object to the image plane (Figure 2.6). As a result, the projected image is always in focus without any distortion [52]. The origin of the camera reference frame $\mathbf{O}_{\text{cam}}$ is placed at the aperture and the image plane is placed a distance f behind the aperture.

[1]Precursor to the *Camera Obscura* described by Aristotle (384 to 322 BC).

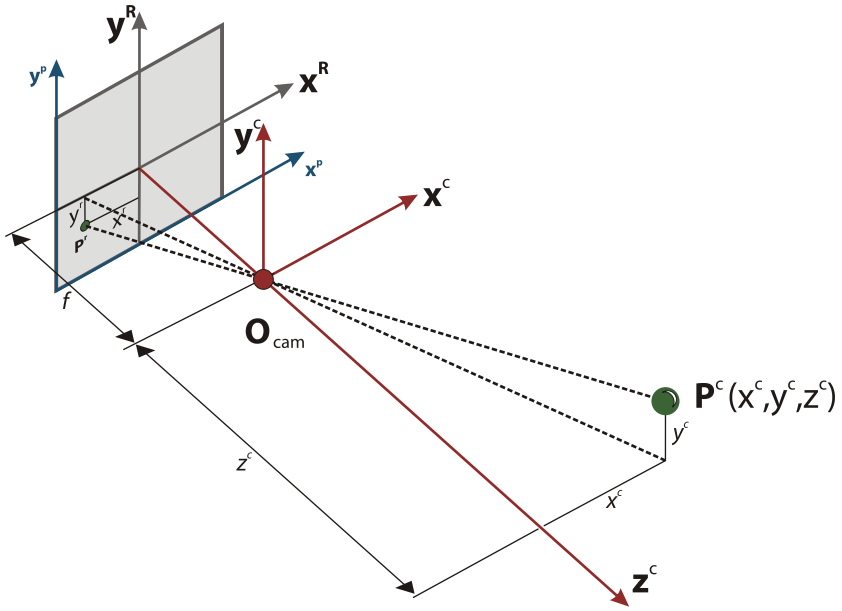


Figure 2.6: Projection of a feature in the camera reference frame to the image plane

Using similar triangles, the perspective projection which relates a 3D object point in the camera reference frame to a 2D point on the real image plane can be written as:

$$\begin{aligned} x^r &= -f \cdot \frac{x^c}{z^c} \\ y^r &= -f \cdot \frac{y^c}{z^c} \end{aligned} \tag{2.1}$$

where f is the camera's focal length. These coordinates can be transformed to the pixel coordinate frame using:

$$\begin{aligned} x^p &= \frac{x^r}{w_{px}} + pp_x \\ &= \frac{-f}{w_{px}} \cdot \left(\frac{x^c}{z^c}\right) + pp_x \\ y^p &= \frac{y^r}{w_{px}} + pp_y \\ &= \frac{-f}{w_{px}} \cdot \left(\frac{y^c}{z^c}\right) + pp_y \end{aligned} \tag{2.2}$$

where w_{px} is the width of a pixel on a digital image sensor and pp_x and pp_y are the coordinates of the sensor's principle point.

In this application, objects of interest are coordinated in the inertial frame (Figure 2.7) and need to be transformed to the camera reference frame before Equation 2.1 can be applied. This transformation consists of two steps: a translation to compensate for the offset of the camera frame's origin in the inertial frame and a rotation to compensate for

the relative attitude of the camera. The combined transformation can be written as:

$$\begin{aligned} \mathbf{P}^C &= \begin{bmatrix} x^c \\ y^c \\ z^c \end{bmatrix} \\ &= \mathbf{DCM}_{I\to C} \cdot (\mathbf{P}^I - \mathbf{O}^I_{cam}) \end{aligned} \tag{2.3}$$

The projection (Equation 2.1) and transformation (Equation 2.3) can be combined to form a single set of equations to project world coordinated objects to the real image plane.

$$\begin{aligned} \mathbf{p}^R &= \begin{bmatrix} x^r & y^r \end{bmatrix}^T \\ &= \begin{bmatrix} -f\frac{x^c}{z^c} & -f\frac{y^c}{z^c} \end{bmatrix}^T \\ &= \begin{bmatrix} \frac{\mathbf{dcm}^{c,x}_w \cdot (\mathbf{P}^I - \mathbf{O}^I_{cam})}{\mathbf{dcm}^{c,z}_w \cdot (\mathbf{P}^I - \mathbf{O}^I_{cam})} & \frac{\mathbf{dcm}^{c,y}_w \cdot (\mathbf{P}^I - \mathbf{O}^I_{cam})}{\mathbf{dcm}^{c,z}_w \cdot (\mathbf{P}^I - \mathbf{O}^I_{cam})} \end{bmatrix}^T \\ &= f\left(\mathbf{x}_{cam}, \mathbf{P}^I\right) \end{aligned} \tag{2.4}$$

with

$$\mathbf{DCM}_{I\to C} = \begin{bmatrix} \mathbf{dcm}^{c,x}_w \\ \mathbf{dcm}^{c,y}_w \\ \mathbf{dcm}^{c,z}_w \end{bmatrix}$$

and

$$\mathbf{x}_{cam} = [\phi_{cam}, \theta_{cam}, \psi_{cam}, N_{cam}, E_{cam}, D_{cam}]^T$$

These coordinates in the real image frame can be converted to discrete pixel coordinates using Equation 2.2.

2.4 Markers

The term markers refers to items that are placed at the intended landing site to provide the VPAM system with enough information to determine the vehicle's position and attitude relative to the inertial frame. The implementation of the markers is discussed in Chapter 5 and this section only considers the number of markers and their configuration.

The system proposed by [92] made use of four markers arranged in a square around the runway. This provided sufficient information to guide a scale aircraft towards the touch-down point. When a square configuration is viewed from above there are four ambiguities in terms of the heading angle due to symmetry. The simplest configuration without this

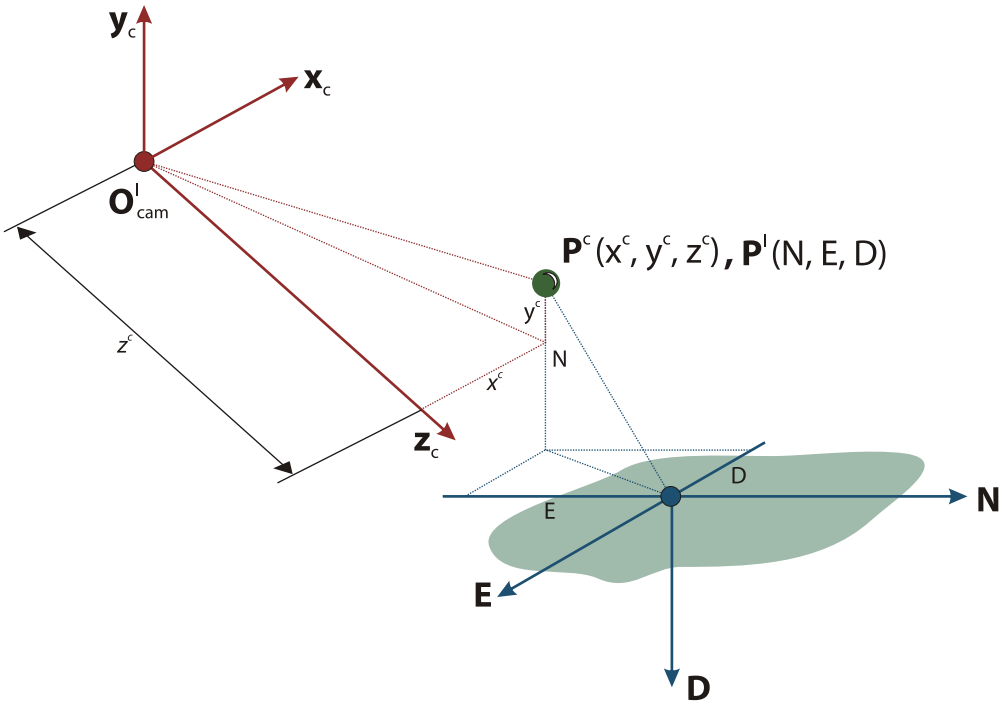


Figure 2.7: Transformation of a world coordinated feature to the camera references frame

symmetry is an isosceles triangle. This configuration, however, suffers from an ambiguity if the camera's attitude changes and could lead to incorrect marker identificaton. Four markers arranged in a rectangular configuration will remove two of the ambiguities, but still won't be able to provide the absolute heading angle. If a fifth marker is added strategically to a square configuration, the ambiguity is removed. The chosen configuration is shown in Figure 2.8.

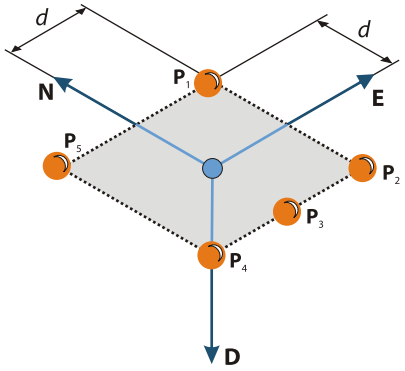


Figure 2.8: Marker configuration

If the positions of $\mathbf{P}_1$ to $\mathbf{P}_5$ in the inertial frame are known, $\left[\mathbf{P}_1^I \dots \mathbf{P}_5^I\right]$, their centroids (projections on the real image frame) can be determined from Equation 2.4:

$$\left[\mathbf{p}_1^R \dots \mathbf{p}_5^R\right] = f\left(\mathbf{x}_{cam}, \left[\mathbf{P}_1^I \dots \mathbf{P}_5^I\right]\right) \tag{2.5}$$

Using Equation 2.2 these projections can be transformed to the discrete pixel frame resulting in $\left[\mathbf{p}_1^P \dots \mathbf{p}_5^P\right]$.

In the derivations to follow, a marker is represented by a single world coordinate which is projected to a single image plane coordinate with no regard for the size of the marker or its projection. Practically, a marker's projection is a region of high intensity pixels. To obtain the single image plane coordinate required by the computer vision algorithms, the region's center of mass (or centroid) is used as an approximation. In the remainder of the document, centroid is used to refer to a single image plane coordinate that represents the projection of a specific marker. Phantom centroid is used to refer to the single image plane coordinate that represents the projection of an undesired projection.

2.5 Correspondence Matching

The correspondence problem is the process required to find a set of points in one image which can be identified as the same points in another image [59] or an actual object. This is a fundamental problem present at the core of a large group of image-based applications such as object tracking [64], object recognition [64, 59], camera calibration [64], structure and motion estimation [99], and stereo vision [59].

2.5.1 Literature Study

Proposed methods to solve the correspondence problem can be divided into two categories [95, 99]: area based methods and feature matching methods.

Area Based Methods

Area based methods (also called correlation-like methods) deal with an image without trying to detect features contained in it. These methods make use of the correlation of the intensity of image patches under the assumption that the patches in the different views contain some similarity [77]. They require the area to be texture rich for this assumption to be valid [95], which makes the algorithm sensitive to contrast changes [46]. Methods to remove this restriction are considered in [46]. Since the correlation needs to be recalculated for multiple regions, area based methods are inherently computationally inefficient [96]. Area based methods are generally sensitive to changes in image scale, rotation, viewpoint and illumination. Gaussian derivatives [22] and Gabor wavelets [41] have been proposed to overcome these limitations at the cost of computational efficiency.

Feature Matching Methods

The second group of methods first extract some features (edge segments, corners, etc) from the image. These features are then organised in a graph with the features defining the nodes and the geometric relationships between the features defining the links [95]. The correspondence problem is now reduced to the registration of the two graphs (or subgraph isomorphism). Common techniques to solve this problem include tree searching [95], relaxation [95] [59], maximal clique detection [95], optimisation methods [35, 20, 64]

and Hopfield networks [59]. These methods are generally more computationally efficient since only a subset of the pixels in the image are used [95, 59]. They might fail, however, if the required features cannot be reliably detected in the images [95].

Feature matching methods rely on a list of low level features extracted from the image before matching can be performed. Typical features used are edges and corners. Edges are regions with an abrupt change in image intensity, normally due to discontinuities in depth, surface orientation, changes in material properties and variations in illumination [77]. Two common categories for edge detection are template matching and differential gradient methods [16]. Both methods are sensitive to varying levels of illumination, while the differential gradient method is more accurate but has a higher computational cost. Template matching is still very widely used if the orientation of the edges are not needed [16]. Corners are defined as a point where an edge changes direction by a large amount [44]. Many different approaches have been considered to detect corners in images: gradient methods, correlation methods, Hough transforms and simpler methods, such as SUSAN (Smallest Univalue Segment Assimilating Nucleus), which does not rely on image derivatives and is less sensitive to noise [16]. All these extraction methods require at least one pass through the image and tend to be very computationally demanding. They are also sensitive to varying levels of illumination which can present robustness problems in the unknown conditions for which the system is designed.

Feature extraction has been cited as one of the main bottlenecks in vision systems [78] where 90% of the processing cycle is spent on low level tasks such as feature extraction. In [99] a system of coded and non-coded reference points are used to simplify the extraction process. Another approach aimed at simplifying the feature extraction stage was mentioned in Section 1.1.2, where red markers were used as feature points. By using only the red layer from the camera output, the feature extraction process is simplified enormously [65].

2.5.2 Proposed Solution

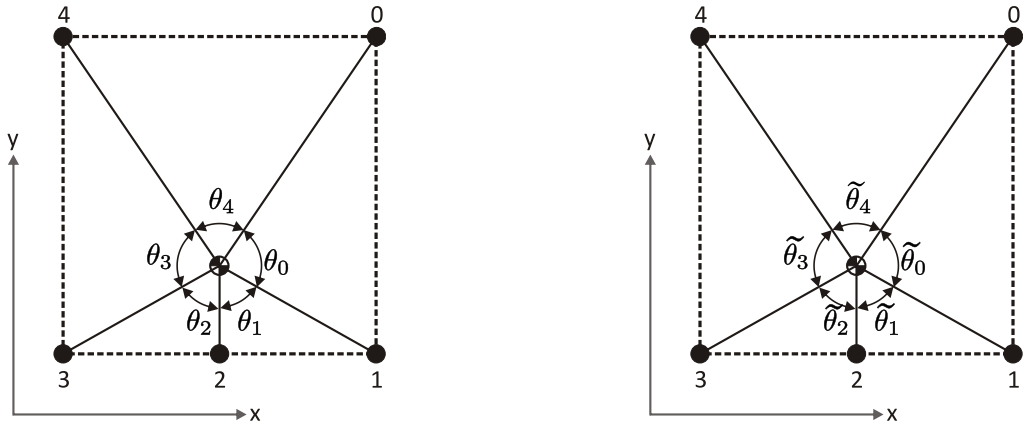
In Section 2.5.1 methods to solve the correspondence problem were outlined with the algorithms requiring one or more traversals of the image with costly operations on individual or groups of pixels. They also exhibited sensitivity to variable illumination conditions which will either decrease the system's robustness or limit its operating envelope. Modifications have been made to improve the robustness [46, 41, 22], but the computational cost is still too high for embedded implementation on low power platforms. If the image is simplifed through the two conditions mentioned in Section 2.1, a feature matching algorithm can be implemented without having to perform feature extraction at a low level. This will allow for an embedded implementation.

The proposed algorithm uses the latest position and attitude information from the onboard estimator and a pinhole camera model to generate a virtual photograph which contains an estimate of the camera's current view of the n markers in the landing tar-

get. Each unique group of n centroids in the actual photograph are then used together with the reference centroids to determine how well the current group matches the virtual image. The quality of the match is calculated using a cost function. After all groups of n centroids have been considered, the combination with the lowest cost is considered the best match. If this cost is lower than a predetermined threshold, the corresponding group is considered a valid match.

Processing the Reference Image

For the latest position and attitude estimate the virtual image can be calculated using a pinhole camera model. The resulting two-dimensional reference centroids are stored in list $\mathcal{C}$.



(a) Reference image (b) Actual image

Figure 2.9: Metric used for correspondence matching

The angles between successive image centroids and the group average $(\bar{p}_x, \bar{p}_y)$ are calculated and stored in Θ:

$$\Theta = [\theta(0), \ldots, \theta(i), \ldots, \theta(n-1)] \tag{2.6}$$

where

$$\theta(i) = \cos^{-1}[\theta_\tau] \tag{2.7}$$

and

$$\theta_\tau = \frac{(\mathcal{C}(i)_x - \bar{p}_x)\,(\mathcal{C}(\mathrm{R}_n^{i+1})_x - \bar{p}_x) + (\mathcal{C}(i)_y - \bar{p}_y)\,(\mathcal{C}(\mathrm{R}_n^{i+1})_y - \bar{p}_y)}{\sqrt{(\mathcal{C}(i)_x - \bar{p}_x)^2 + (\mathcal{C}(i)_y - \bar{p}_y)} \cdot \sqrt{(\mathcal{C}(\mathrm{R}_n^{i+1})_x - \bar{p}_x)^2 + (\mathcal{C}(\mathrm{R}_n^{i+1})_y - \bar{p}_y)}}$$

and

$$(\bar{p}_x, \bar{p}_y) = \left(\frac{1}{n} \sum_{i=0}^{n-1} \mathcal{C}(i)_x, \frac{1}{n} \sum_{i=0}^{n-1} \mathcal{C}(i)_y \right) \tag{2.8}$$

where R_n^i represents the remainder of the quotient of i and n.

Processing the Actual Image

From the list of m centroids obtained from the camera ($\tilde{\mathcal{C}}$), a unique list of n centroids are chosen and placed in $\tilde{\mathcal{C}}_s$. If m is smaller than n, the pattern is not suitable for pose detection and is rejected. If m is larger than (or equal to) n, the following steps will be repeated $\binom{m}{n}$ times:

1. The current group of centroids must be arranged in order such that the angle between successive centroids and the group average can be calculated. This can be achieved by calculating the group average of the selected centroids and determining the angular displacement of each centroid from an arbitrary axis through the group average (Figure 2.10). The angular displacements are stored in a list (Ψ). By sorting Ψ and making the same exchanges on the centroids stored in $\tilde{\mathcal{C}}_s$ the centroids are placed in sequence.

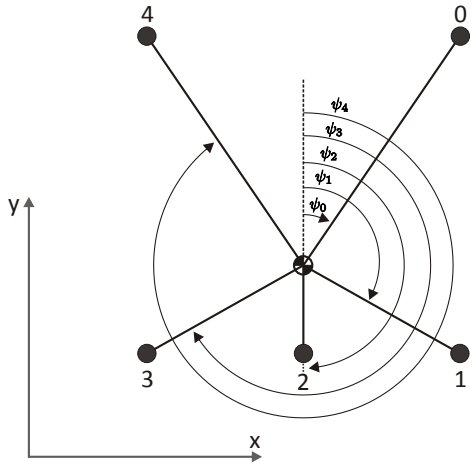


Figure 2.10: Geometry for arranging the current centroid group.

The angular displacement list is given by:

$$\Psi = [\psi(0), \ldots, \psi(i), \ldots, \psi(n-1)] \tag{2.9}$$

where

$$\psi(i) = \begin{cases} \arccos\left[\frac{\tilde{\mathcal{C}}_s(i)_y - \bar{p}_y}{\sqrt{(\tilde{\mathcal{C}}_s(i)_x - \bar{p}_x)^2 + (\tilde{\mathcal{C}}_s(i)_y - \bar{p}_y)^2}}\right] & \tilde{\mathcal{C}}_s(i)_x \geq \bar{p}_x, \\ 2\pi - \arccos\left[\frac{\tilde{\mathcal{C}}_s(i)_x - \bar{p}_x}{\sqrt{(\tilde{\mathcal{C}}_s(i)_x - \bar{p}_x)^2 + (\tilde{\mathcal{C}}_s(i)_y - \bar{p}_y)^2}}\right] & \tilde{\mathcal{C}}_s(i)_x < \bar{p}_x \end{cases}$$

and

$$(\bar{p}_x, \bar{p}_y) = \left(\frac{1}{n}\sum_{i=0}^{n-1} \tilde{\mathcal{C}}_s(i)_x, \frac{1}{n}\sum_{i=0}^{n-1} \tilde{\mathcal{C}}_s(i)_y\right) \tag{2.10}$$

Since n is generally small and partially in order (due to how the image is received from the camera), an insertion sorting algorithm is used to sort Ψ (and $\tilde{\mathcal{C}}_s$)) [74].

2. With the centroids of the current group sorted and stored in $\tilde{\mathcal{C}}_s$, the angles between successive image centroids and the group average $(\bar{p}_x, \bar{p}_y)$ are calculated and stored

in $\tilde{\Theta}_s$:

$$\tilde{\Theta}_s = [\theta(0), \ldots, \theta(i), \ldots, \theta(n-1)] \tag{2.11}$$

where

$$\theta(i) = \cos^{-1}[\theta_\tau] \tag{2.12}$$

and

$$\theta_\tau = \frac{\left(\tilde{\mathcal{C}}_s(i)_x - \bar{p}_x\right)\left(\tilde{\mathcal{C}}_s(\mathrm{R}_n^{i+1})_x - \bar{p}_x\right) + \left(\tilde{\mathcal{C}}_s(i)_y - \bar{p}_y\right)\left(\tilde{\mathcal{C}}_s(\mathrm{R}_n^{i+1})_y - \bar{p}_y\right)}{\sqrt{\left(\tilde{\mathcal{C}}_s(i)_x - \bar{p}_x\right)^2 + \left(\tilde{\mathcal{C}}_s(i)_y - \bar{p}_y\right)} \cdot \sqrt{\left(\tilde{\mathcal{C}}_s(\mathrm{R}_n^{i+1})_x - \bar{p}_x\right)^2 + \left(\tilde{\mathcal{C}}_s(\mathrm{R}_n^{i+1})_y - \bar{p}_y\right)}}$$

and

$$(\bar{p}_x, \bar{p}_y) = \left(\frac{1}{n}\sum_{i=0}^{n-1}\tilde{\mathcal{C}}_s(i)_x, \frac{1}{n}\sum_{i=0}^{n-1}\tilde{\mathcal{C}}_s(i)_y\right) \tag{2.13}$$

3. After sorting the centroids there are still n possible combinations that can be matched to the reference centroids. Each combination has the same sequence, but with a different starting centroid. Each of the combinations can be obtained by simply shifting all the elements of $\tilde{\Theta}_s$ in a circular fashion before the cost is determined:

$$\hat{\Theta}_s^i = \left[\tilde{\Theta}_s(\mathrm{R}_n^i), \tilde{\Theta}_s(\mathrm{R}_n^{i+1}), \ldots, \tilde{\Theta}_s(\mathrm{R}_n^{i+(n-1)})\right] \quad i \in 0 \ldots n-1 \tag{2.14}$$

where i is the index of the current starting centroid.

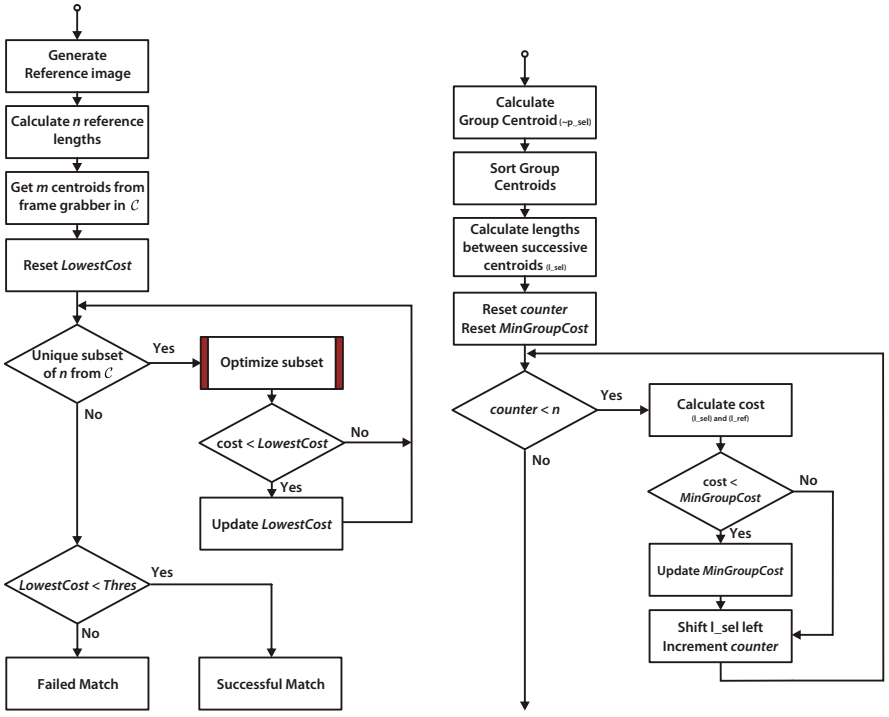
4. The cost function is a simple sum of squares of the deviation between the metrics of the actual centroids and the reference image. Intuitively this means that, if the orientation and spacing of the actual image differs a lot from the reference image, the cost will be high. Conversely, a good match will be represented by a low cost. For the current group of n centroids, the minimum cost for the current group of centroids can be calculated by considering each possible combination:

$$\mathcal{J}_\Theta = \min\left[\left\{j_\Theta : j_\Theta = \sum_{k=0}^{n-1}\left|\hat{\Theta}_s^i(k) - \Theta(k)\right|, \quad i \in [0 \ldots n-1]\right\}\right] \tag{2.15}$$

5. Steps 1 to 4 are repeated for all $\binom{m}{n}$ combinations of n centroids. During the first iteration $\mathcal{J}_\Theta$ is considered to be the minimum overall cost. During each subsequent iteration the latest $\mathcal{J}_\Theta$ is compared to the current minimum overall cost. If the latest cost is less than the current minimum overall cost, the minimum overall cost is updated. When all $\binom{m}{n}$ combinations have been considered, the best fit will be represented by the minimum overall cost ($\hat{\mathcal{J}}_\Theta$). If $\hat{\mathcal{J}}_\Theta$ is smaller than a predetermined threshold, a valid match has been achieved. Details of this threshold are provided in Section 2.5.3.

Implementation

The flow of the algorithm is shown in Figures 2.11a and 2.11b.



(a) Overview (b) Overview of subset optimization

Figure 2.11: Flow for correspondence matching

2.5.3 Threshold Determination

To determine whether the combination leading to the minimum cost is valid, the cost is compared to a threshold. This threshold must be chosen to prevent or reduce invalid matches from being accepted and prevent or reduce valid matches from being rejected. A simple approach is followed where the matching algorithm is performed at a number of discrete points within the operating window of the system and by introducing phantom centroid in the true image while sweeping the threshold at each test point. When analysing the matching result, the number of false rejections and invalid matches (as a function of threshold and altitude) can be determined.

The matching algorithm is performed a number of times for each discrete lateral position (n, e) in each discrete altitude step h_{step} bounded by a cylinder (radius R, height h, height offset h_0) centered at the landing target (Figure 2.12). The lateral positions are determined by the cylinder radius R, a discrete radial step r_{step} and the sweep angle θ (Figure 2.13a):

$$n = (R + i \cdot r_{\text{step}}) \cdot \cos((j-1)\theta) \tag{2.16}$$

$$e = (R + i \cdot r_{\text{step}}) \cdot \sin((j-1)\theta) \tag{2.17}$$

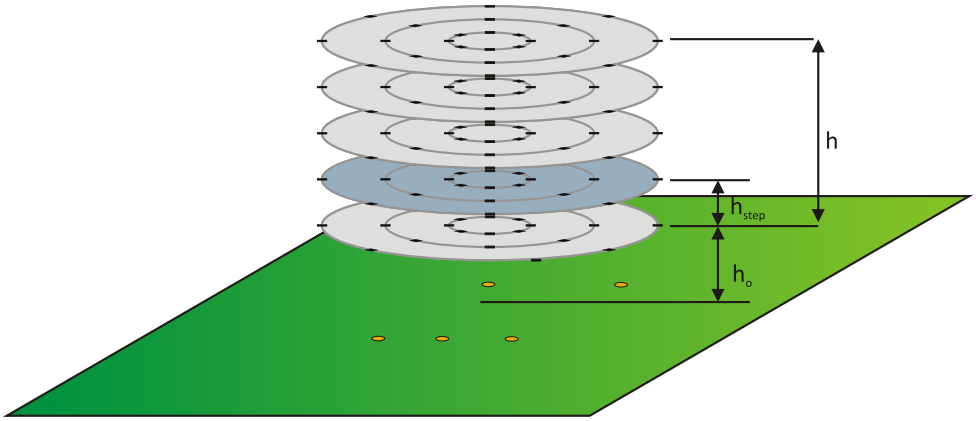
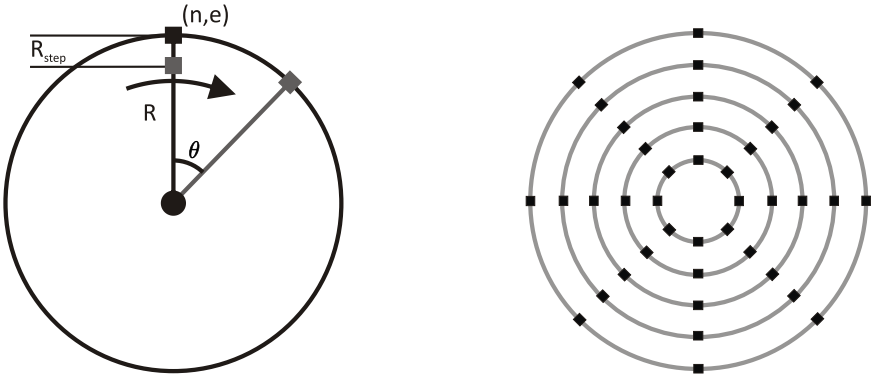


Figure 2.12: Iterations during the threshold determination

where i is a counter that controls the radial iteration and j is a counter that controls the angular iteration.



(a) Discrete lateral position iteration

(b) Discrete lateral position result

Figure 2.13: The discrete iteration of the lateral position

At each of these positions, for a specific sequence of threshold values, the matching algorithm is performed a number of times (n) using a true image generated from the current position and attitude (corrupted with random phantom centroids) and a state estimate generated by adding a random (σ_{pos} and σ_{att}) delta vector to the true state. After the completion of each altitude step the matching result is compared to the expected result to generate the false-rejection and invalid-match statistics for varying threshold at that altitude.

Results

The algorithm was performed with the parameters in Table 2.1 and the results are shown in Figures 2.14 and 2.15.

Table 2.1: Parameters used for matching threshold calculation

Parameter	Value	Notes
R	3 m	Radius
h	6 m	Cylinder height
h_0	2 m	Cylinder offset, altitude range: $h_0 - (h_0 + h)$
h_{step}	0.05 m	Altitude step
r_{step}	0.2 m	Radial distance step
θ	20°	Sweep angle
n	100	Number of iterations per threshold value at each lateral position
σ_{pos}	3 m	Standard deviation for position estimate
σ_{att}	5°	Standard deviation for attitude estimate
T_{max}	100	Maximum treshold value
T_{step}	0.01	Treshold increment value

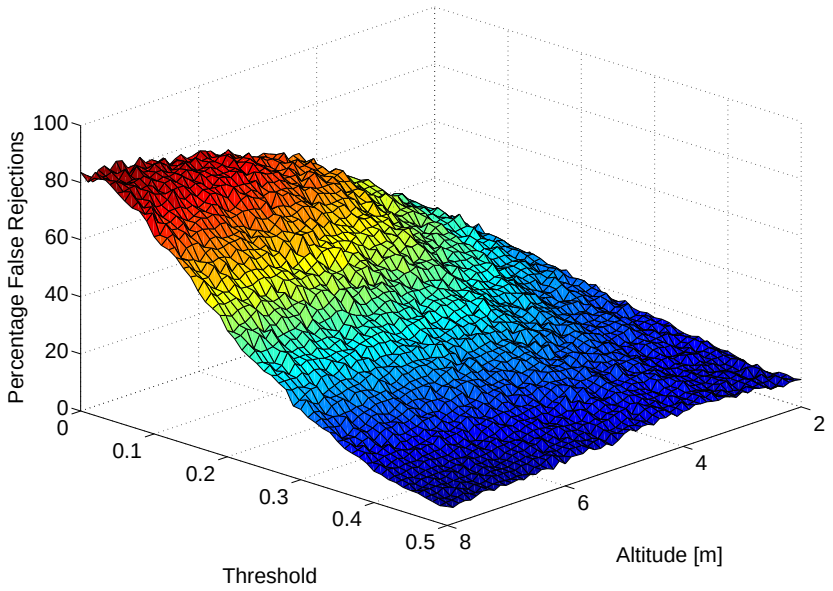


Figure 2.14: Percentage of false rejections as a function of threshold and altitude

From Figures 2.14 and 2.15 the following observations can be made:

1. As the threshold increases, the number of false rejections decreases since a larger deviation between the true and virtual image can be tolerated.

2. As the threshold increases, the number of invalid matches increases since the possibility for a phantom centroid leading to a slightly lower cost increases.

3. Invalid matches will not result in catastrophic failure of the system since these matches are typically caused by a false marker situated close to the desired marker. H will, however, lead to a decrease in accuracy.

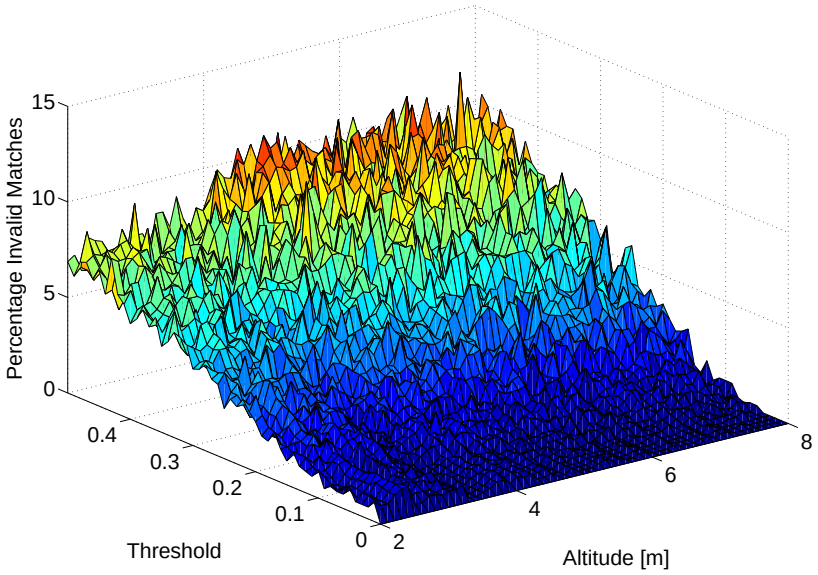


Figure 2.15: Percentage of invalid matches as a function of threshold and altitude

4. The design decision for the matching threshold is reduced to a decision of how many rejections of valid matches can be tolerated while the number of accepted invalid matches must be kept in check.

5. There is an increase in reliability in terms of both false rejections and invalid matches with decreasing altitude. This an artifact of the manner in which the false markers are introduced to simulate reflections from real world objects. The false markers are created by adding a random position shifts to duplicates of the markers in world coordinates. This method simulates the effect of false markers introduced by disturbances such as reflective surfaces and not camera non-idealities (such as internal reflections and light leakages). At higher altitude, the projection of this marker will be closer to the actual marker's projection and could cause matching problems. At lower altitudes, the projection is spread further and does not interfere with the matching.

2.5.4 Simulation

The simulation was created to extensively test the correspondence matching algorithm under a number of extreme circumstances. These include the occlusion of one or more markers (Experiment 1), the introduction of one or more phantom centroids (Experiment 2) and a combination of both (Experiment 3).

Experiment 1: Partial Occlusion

Occlusions are the obstruction of one or more of the landing target markers due to another object moving between the camera and the marker, or the marker moving out of the field of

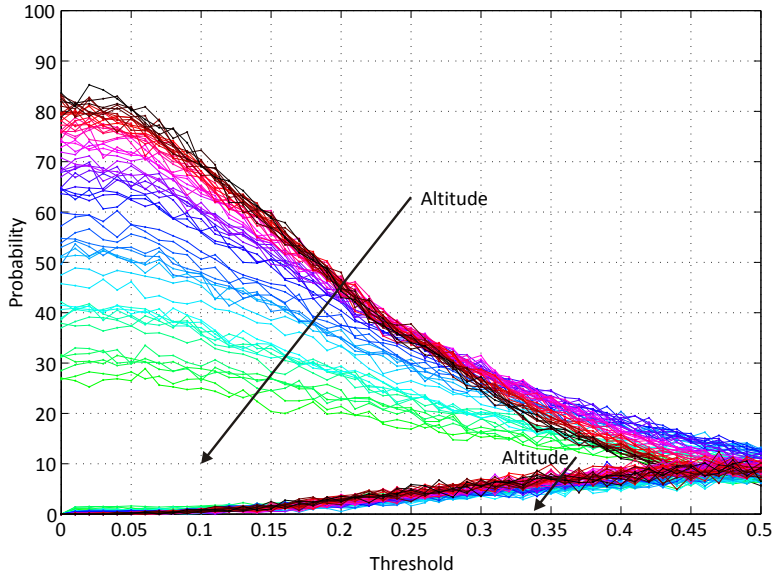


Figure 2.16: Percentage of invalid matches and false rejections as function of threshold

view. If there are only occlusions (and no false targets), there will be less than n centroids in the resulting image. The matching algorithm will not process such an image and thus prevent inaccurate information from being reported by the subsequent vision algorithms. To prove the functionality, however, tests having only occlusions were performed.

In Experiment 1, a random number of centroids (between 1 and n) were occluded at a time. The experiment contained 100 000 trials. The result was a successful match percentage of 0% which is the desired result.

Experiment 2: Phantom Centroids

Phantom centroids are markers appearing in the image that are not part of the landing target. These will have reduced intensity when compared to the actual markers, and may be eliminated during the thresholding stage if the intensity is attenuated enough. If they are not elimated during thresholding, the matching algorithm is responsible for determining the valid selection of centroids.

In Experiment 2, a random number of phantom centroids (between 1 and n) were introduced in the image while no targets were occluded. The experiment contained 100 000 trials. From the results presented in Table 2.2 it is clear the the bounds in Figure 2.16 is adequate to estimate the number of false rejections and invalid matches.

Experiment 3: Occlusion and Phantom Centroids

When a phantom centroids is introduced while one or more targets are occluded it will not always be detected during the first stage of the matching algorithm since the total number of centroids could be larger than n. The optimisation is then responsible for rejecting the image.

Table 2.2: Results of matching Experiment 2

Threshold	Invalid Matches	Percentage	False Rejection	Percentage
0.1	80	0.08	60251	60.25
0.2	45	0.05	22825	22.83
0.3	697	0.70	6897	6.90
0.4	869	0.87	2295	2.30
0.5	1000	1.00	1008	1.01

In Experiment 3, a random number of centroids were occluded and replaced by at least the same number of false centroids to result in a maximum of eight centroids. The experiment contained 100 000 trials. The results are shown in Table 2.3 and shows a small percentage of invalid matches and no false rejections. The false rejection result is correct since there could be no false rejections if no correct matches were expected (due to the occlusions). The invalid match result required some investigation since the result implies that a match was validated by the algorithm while it should not have been (again due to the occlusion). The explanation is clear from Figure 2.17: the randomly placed phantom centroid was placed within tolerance of the occluded marker. As a result, the cost calculated by the matching algorithm was below the threshold and the match was accepted. The consequence is a decrease in accuracy of the pose estimation algorithm.

Table 2.3: Results of matching Experiment 3

Threshold	Invalid Matches	Percentage	False Rejection	Percentage
0.1	11	0.01%	0	0%
0.2	109	0.11%	0	0%
0.3	274	0.27%	0	0%
0.4	941	0.94%	0	0%
0.5	1871	1.87%	0	0%

Conclusion

From the three experiments outlined in the previous section the validity and robustness of the matching algorithm has been confirmed.

2.6 Pose Estimation

The pinhole camera model is used to project features in the inertial frame to the real image frame or the pixel coordinate frame. The result of this projection is an image that shows what a photograph would have looked like if taken from a certain relative position and attitude. For a camera to be used as a position and attitude sensor, the opposite process is required. Given a two dimensional image and a set of known points in the inertial frame the position and attitude of the camera relative to the inertial frame at the

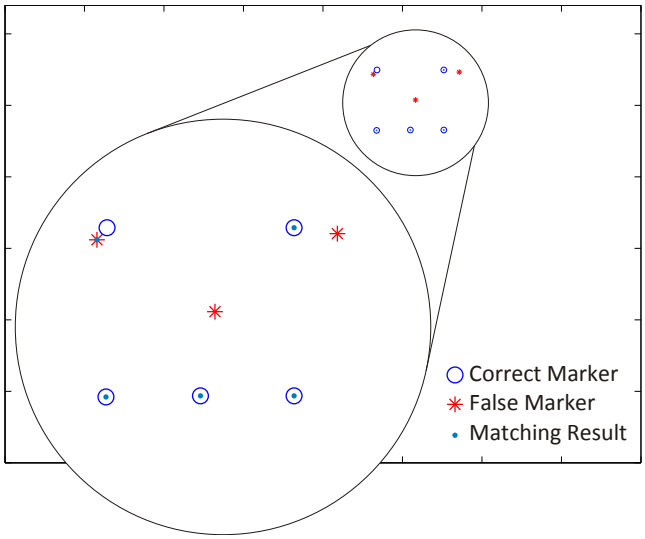


Figure 2.17: Image that resulted in a valid match despite an occlusion

instant the photo was taken, must be determined. This is known as the pose estimation problem in computer vision and has received considerable attention in the last three decades [89, 47, 94, 49, 73, 38, 66, 15, 62]. One of the most widely used methods involve the minimisation of the reprojection error to determine an approximation to the camera state vector. Due to the iterative nature of this algorithm it is not ideally suited for embedded implementation. In this section the iterative optimisation method is critically evaluated (Section 2.6.1 and a more efficient alternative is outlined (Section 2.6.2). A detailed comparison between the optimisation and non-iterative methods is discussed in Section 2.6.3.

2.6.1 Optimisation Approach

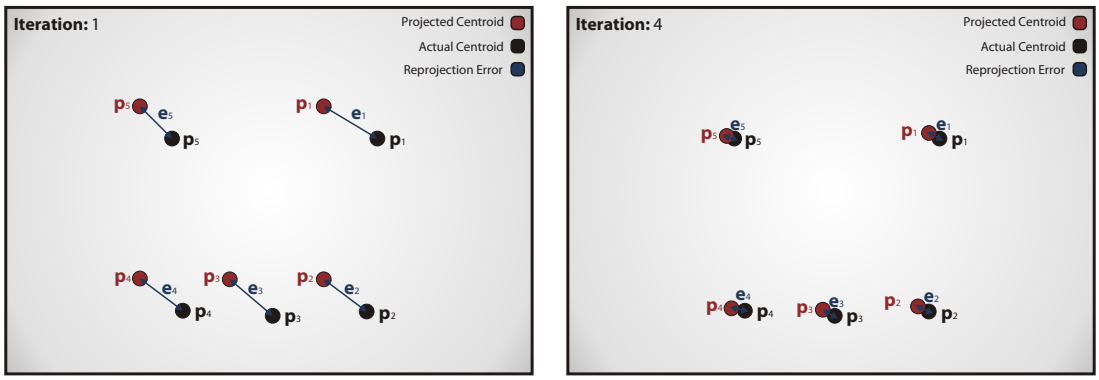
The iterative approach to pose estimation involves the minimisation of the error between the centroids in an actual photograph and the centroids in a virtual image generated using an approximation of the camera's true state vector (position and attitude). When the error is minimised (and the problem has converged to the correct minimum) the state vector will be a very good estimate of the camera's position and attitude at the moment in time when the actual photograph was captured. This section describes the development of the method, the optimisation techniques and a number of performance properties (runtime, parameter sensitivity, robustness against noise and convergence).

Cost Function

A cost function based on the sum of squared errors between centroids of the actual and virtual images us used due to intuitive physical interpretation of the cost function [92], as well as the number of optimisation methods for cost functions of that nature:

$$F = \sum_{i=1}^{5} (x_i^r - \widehat{x}_i^r)^2 + (y_i^r - \widehat{y}_i^r)^2 \tag{2.18}$$

where x_i^r and y_i^r are the coordinates of the projection of $\mathbf{P}_i^I$ on the image plane and $\widehat{x}_i^r$ and $\widehat{y}_i^r$ are the coordinates of the model based projection of $\mathbf{P}_i^I$. By adjusting the camera state vector in an iterative manner, the cost function in Equation 2.18 can be minimised (Figure 2.18). The camera state vector that minimises the cost function is an approximation of the camera's pose when the photo was taken.



(a) First iteration (b) Fourth iteration

Figure 2.18: Examples of two iterations where the reprojection error is minimised to determine an approximation of the camera state.

A minimum for this cost function would imply that the actual marker centroids and the projected centroids are very close to each other and that the estimated camera state vector is a good approximation of the actual camera states. The state estimate from the Kalman filter that is maintained on the OBC is used as the initial state vector in the optimisation process.

The cost function can be written as a vector function, allowing it to be used in a wide range of optimisation techniques [51]:

$$\begin{aligned} F(\mathbf{x}_{cam}) &= \frac{1}{2}\sum_{i=1}^{m}\left[f_i(\mathbf{x}_{cam})\right]^2 \\ &= \frac{1}{2}\left|\mathbf{f}(\mathbf{x}_{cam})\right| \end{aligned} \tag{2.19}$$

where $\mathbf{f}$ is a vector function relating $\mathbb{R}^n \mapsto \mathbb{R}^m$ with $m \geq n$. The cost function in Equation 2.18 can be written as:

$$\begin{aligned} F(\mathbf{x}_{cam}) &= \frac{1}{2}\sum_{i=1}^{10}\left[f_i(\mathbf{x}_{cam})\right]^2 \\ &= \frac{1}{2}\left|\mathbf{f}(\mathbf{x}_{cam})\right| \\ &= \frac{1}{2}\mathbf{f}(\mathbf{x}_{cam})^T\mathbf{f}(\mathbf{x}_{cam}) \end{aligned} \tag{2.20}$$

with

$$\mathbf{f}(\mathbf{x}_{cam}) = \left[f_1(\mathbf{x}_{cam}), \ldots, f_i(\mathbf{x}_{cam}), \ldots, f_{10}(\mathbf{x}_{cam})\right] \tag{2.21}$$

and

$$f_i(\mathbf{x}_{cam}) = \begin{cases} \left(p_x^{\lceil i/2 \rceil} - \widehat{p}_x^{\lceil i/2 \rceil}\right) & , i \text{ odd} \\ \left(p_y^{\lceil i/2 \rceil} - \widehat{p}_y^{\lceil i/2 \rceil}\right) & , i \text{ even} \end{cases} \tag{2.22}$$

and

$$\mathbf{x}_{cam} = [\phi_{cam}, \theta_{cam}, \psi_{cam}, N_{cam}, E_{cam}, D_{cam}]^T$$

The derivatives of $F(\mathbf{x}_{cam})$ can be expressed in terms of the Jacobian of $\mathbf{f}$, which is the $m \times n$ matrix of first partial derivatives:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial \phi} & \frac{\partial f_1}{\partial \theta} & \frac{\partial f_1}{\partial \psi} & \frac{\partial f_1}{\partial N} & \frac{\partial f_1}{\partial E} & \frac{\partial f_1}{\partial D} \\ \frac{\partial f_2}{\partial \phi} & \frac{\partial f_2}{\partial \theta} & \frac{\partial f_2}{\partial \psi} & \frac{\partial f_2}{\partial N} & \frac{\partial f_2}{\partial E} & \frac{\partial f_2}{\partial D} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_{10}}{\partial \phi} & \frac{\partial f_{10}}{\partial \theta} & \frac{\partial f_{10}}{\partial \psi} & \frac{\partial f_{10}}{\partial N} & \frac{\partial f_{10}}{\partial E} & \frac{\partial f_{10}}{\partial D} \end{bmatrix} \tag{2.23}$$

The gradient and Hessian[1] of F can be determined as [40]:

$$\mathbf{g} = \nabla f(\mathbf{x}_{cam}) = \mathbf{J}(\mathbf{x}_{cam})^T \mathbf{f}(\mathbf{x}_{cam}) \tag{2.24}$$

$$\mathbf{H} = \nabla^2 f(\mathbf{x}_{cam}) = \underbrace{\mathbf{J}(\mathbf{x}_{cam})^T \mathbf{J}(\mathbf{x}_{cam})}_{H_1} + \underbrace{\sum_{j=1}^{10} \mathbf{f}_j(\mathbf{x}_{cam}) \nabla^2 \mathbf{f}_j(\mathbf{x}_{cam})}_{H_2} \tag{2.25}$$

The group of iterative algorithms that are considered in Section 2.6.1 only calculate the first derivative of the residuals. When the Hessian is required it is approximated from Equation 2.25:

$$\begin{aligned} \mathbf{H} = \nabla^2 f(\mathbf{x}_{cam}) &= \mathbf{J}(\mathbf{x}_{cam})^T \mathbf{J}(\mathbf{x}_{cam}) + \sum_{j=1}^{10} \mathbf{f}_j(\mathbf{x}_{cam}) \nabla^2 \mathbf{f}_j(\mathbf{x}_{cam}) \\ &\approx \mathbf{J}(\mathbf{x}_{cam})^T \mathbf{J}(\mathbf{x}_{cam}) \end{aligned} \tag{2.26}$$

This approximation is only valid when the residuals $f_i(\mathbf{x}_{cam})$ are very small and/or the cost function is almost linear ($\nabla^2 \mathbf{f}_j(\mathbf{x}_{cam}) \approx 0$) near the solution [40].

Minimization Methods

In [92] the Levenberg Marquadt method was used due to the prevelance of the algorithm in computer vision applications. There are simpler methods available that could be more computationally efficient and this section provides a very brief overview of these methods.

[1] A matrix of second order partial derivatives indicating local curvature

The cost function developed in the previous section (Equation 2.18) is in a form that can be solved by non-linear least-squares optimisation algorithms [51]. The most basic method is the gradient descent optimisation method and thus provides the foundation for other non-linear least-squares solvers. It attempts to minimize a cost function by taking steps proportional to the negative of the gradient at the current point in the cost function. When the cost reaches a certain threshold, the optimisation is complete and the last state value is deemed the minimiser. It is rarely used without modification since the algorithm is generally slow [51, 7].

With a gradient descent optimisation method only the first-order derivatives are used to determine the search direction. The classic Gauss-Newton method (Algorithm 1) is based on the gradient descent algorithm with the only difference being the method in which the search direction is calculated. The minimiser for Equation 2.19 can be found by solving Equation 2.27 for $\mathbf{h}_{gn}$ [51]:

$$\left(\mathbf{J}^T\mathbf{J}\right)\mathbf{h}_{gn} \quad = \quad \text{-}\mathbf{J}^T\mathbf{f} \tag{2.27}$$

The vector $\mathbf{h}_{gn}$ represents the optimal search direction for the current iterate, $\mathbf{x}_{cam}(k)$. This minimizer is both unique and a descent direction for Equation 2.20 [51]. As a result, it can be used as the search direction in the standard descent algorithm to update the current iterate. In the classic Gauss-Newton method a step size of $\alpha = 1$ is taken in all steps.

Algorithm 1 Gauss-Newton Algorithm

1: $k \leftarrow 0$
2: $\mathbf{x}^{cam} \leftarrow \mathbf{x}_0$
3: $found \leftarrow$ **false**
4: **while** (**not** $found$) **and** $(k < k_{max})$ **do**
5: Calculate the search direction $\mathbf{h}_{gn}$ using Equation 2.27.
6: **if** No such $\mathbf{h}_{gn}$ exists **then**
7: $found \leftarrow$ **true**
8: **else**
9: Set the step length α as 1.
10: Perform the step: $\mathbf{x}_{cam}^{k+1} \leftarrow \mathbf{x}_{cam}^{k} + \alpha\mathbf{h}_{gn}$
11: $k \leftarrow k + 1$
12: **end if**
13: **end while**

This approximation is only valid when the residuals $f_i(\mathbf{x}_{cam})$ are very small and/or the cost function is almost linear ($\nabla^2\mathbf{f}_j(\mathbf{x}_{cam}) \approx 0$) near the solution [40]. The small residual constraint can also be interpreted as the case where the initial value of the iteration is close to the true value. The classic Gauss-Newton algorithm has linear convergence in general, with near quadratic convergence when the iterate is close to the solution [51, 19]. Furthermore, if the cost function has neglegible curvature at the solution the final convergence will be near quadratic [40]. Despite this, local convergence is not guaranteed

since the the step size is fixed and the update to the iterate can cause an increase in the cost function value which does not satisfy the descending condition $F_{k+1} \leq F_k$ [51] even though the search direction is a descent direction.

The classic Gauss-Newton method provides a simple method to determine the step direction of the innovation for the next iterate. Since the step size is arbitrarily chosen as 1, local convergence is not guaranteed [40]. Since divergence or the convergence to another minimum (which represents a different solution) is undesired, the method can be augmented with a line search algorithm (golden section search and Amijo's rule [7]) that is used to determine the optimal step length once the step direction has been determined. Details about the line search algorithm will not be repeated here and can be found in [7, 51]. The Gauss-Newton method using a line search is locally convergent [51]. The Gauss-Newton method (both classic and augmented) suffer from a linear convergence rate when the best estimate is far from the solution. As this current iterate approaches the solution, the convergence rate approaches a quadratic rate.

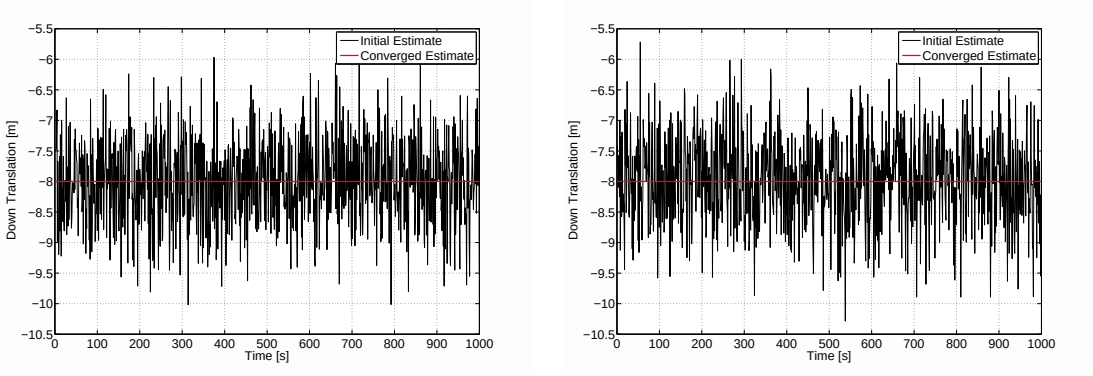
A further modification to this group of methods was done by Levenberg and Marquardt [48]. Their method could be seen as a combination of the gradient descent and Gauss-Newton methods: when the solution is far from the desired minimum it behaves like the gradient method (slow, but guaranteed convergence) and as it approaches the minimum it behaves like the Gauss Newton method (fast convergence) [51]. As mentioned earlier, this method was used in the system developed by [92] and will be used for comparitive purposes in the remainder of this document. It will, however, not be discussed since it as was never considered for the system developed in this project due to the high computational cost. A complete discussion of the algorithm is given in [48, 51].

Convergence

In the previous section it was stated that the augmented Gauss-Newton and Levenberg Marquadt methods have guaranteed local convergence. This, however, does not guarantee convergence to the desired solution since both methods are only valid for the small residual case. The following simulations investigate the convergence properties of the augmented Gauss Newton and Levenberg Marquadt with random initial state estimates.

In the first simulation, estimates with variance similar to the variance of the kinematic state estimator developed by [92] have been used as the initial value to the two optimisation methods. The solution vector was zero attitude and horizontal translation of 2 m in both directions at an altitude of 8 m. The resulting altitude state and the corresponding initial value for both methods are shown in Figure 2.19. This simulation showed that if the initial estimate is close enough to the solution, convergence to that solution is very likely. The same behaviour is present in the other states and is, as a result, not shown.

In the second simulation, the variance of the initial values was increased to larger, but still realistic values that could be received from the state estimator. The solution vector was zero attitude and horizontal translation of 2 m in both directions at an altitude of 8 m. The resulting altitude state and the corresponding initial value for both methods

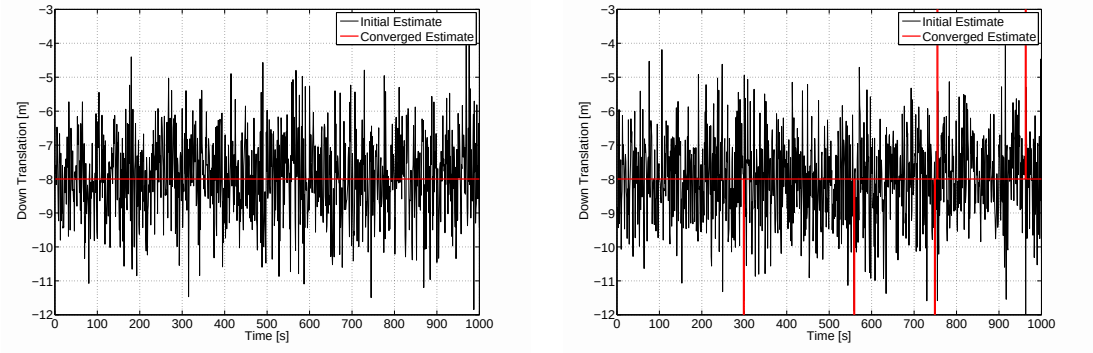


(a) Gauss-Newton with line search (b) Levenberg Marquadt

Figure 2.19: Convergence results for $(\mathbf{x}_{true})_{down}$ = -8 with an initial estimate position variance 0.7m and angular variance of 0.25°

.

are shown in Figure 2.20. Convergence to another minimum is clear for the Levenberg Marquadt method. Similar behaviour is present in the other states but is not shown.



(a) Gauss-Newton with line search (b) Levenberg Marquadt

Figure 2.20: Convergence results for $(\mathbf{x}_{true})_{down}$ = -8 with an initial estimate position variance 1.2m and angular variance of 0.5°

.

From the above results, the functionality of the iterative pose estimation methods are shown, but the risk of convergence to another, undesired, solution is clear. Since iteration and recalculation of the Jacobian is required to recover the approximate camera states, the methods are not very efficient for embedded implementatation. Furthermore, they are dependent on reasonably accurate initial state estimates to function.

2.6.2 Non-Iterative Approach

In the previous section the functionality of the iterative pose estimation methods was confirmed, but their shortcomings in terms of efficiency and robustness were also noted. An alternative approach is described in this section that addresses the problems identified in the previous section. This approach is based on the idea that if a point's coordinates

are known in two reference frames, the relative orientation and translation between the frames can be determined. The marker locations are known in the inertial frame and their projections are known on the image frame while the orientation and position of the camera frame is desired. If the coordinates of each marker can be determined in the camera frame, the orientation and position of the camera frame relative to the inertial frame can be determined. Non-iterative pose estimation algorithms are divided into two parts. The first part, called depth recovery, is responsible for determining the coordinates of the markers in the camera frame given the projections in the image frame. The second part, referred to as frame transformation, is responsible for determining the relative orientation and position between the camera and inertial frames given the coordinates of the markers in both frames.

Sylvester resultants can be used to set up an over-constrained system of fourth order polynomials to perform the depth recovery [66]. The system can be solved using numerical methods. Once the points are known in both the camera frame and the inertial frame, the pose and position of the camera frame can be determined through an analytical quaternion based method [36].

A vector based analytical solution to the depth recovery problem for five non-coplanar markers and four coplanar markers is discussed in [25]. This approach involves the representation of the difference between ray vectors in terms of basis vectors constructed using the unknown ray vectors. These difference vectors, coordinated in the camera frame, represent the vectors between markers. This leads to a system of equations with two unknowns (the ray lengths) per equation. Since vector lengths remain unchanged during frame transformation, another equation could be written in terms of the actual distance between markers and two ray lengths. This equation and the original equation, from the difference vector, that has the same unknowns can be solved simultaneously. The remaining equations are then reduced to single variable equations which can be solved analytically.

An algorithm that uses an iterative least squares method to perform the depth recovery and the singular value decomposition (SVD) algorithm to determine the camera pose and position is discussed in [62]. The SVD based algorithm is outlined in [8]. Various other methods not based on optimisation algorithms have been used to solve the pose estimation problem. These include Kalman filter methods [15, 52]) and neural network methods [85]. Since these methods are generally not computationally efficient they are not considered for this project.

The method outlined in the rest of this section is a combination of the analytical depth recovery method [25] and the SVD-based frame transformation algorithm [8] to perform the transformation.

Depth Recovery

Depth recovery is the calculation of the ray lengths measured from the camera frame's origin (optical centre) to each marker coordinated in the camera frame ($\mathbf{P}_i^C,\ \ i \in 1, \ldots, 5$). The depth recovery procedure is based on the method proposed by [25] for the non-coplanar case. For this derivation, the image plane is placed in front of the optical center at $z^c = f$.

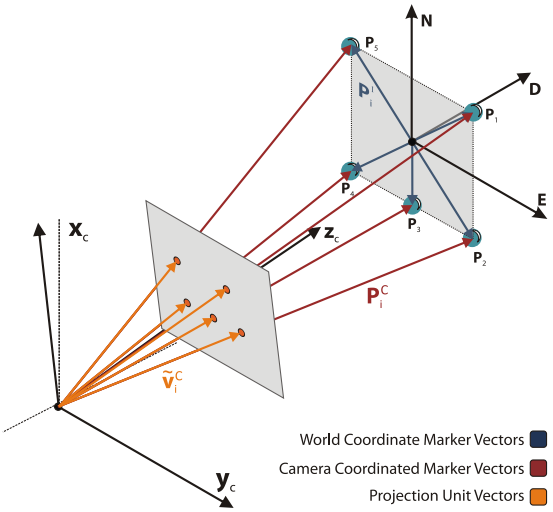


Figure 2.21: Vector definitions for the analytic pose estimation method

A vector to the projection of $\mathbf{P}_i^C$ on the image frame at $z^c = f$ (Figure 2.21) can be written as:

$$\mathbf{v}_i^C = \begin{bmatrix} x_i^r \\ y_i^r \\ f \end{bmatrix} \qquad i \in (1, \ldots, 5) \tag{2.28}$$

where x_f^i and y_f^i are obtained from the matching algorithm.

Since each $\mathbf{P}_i^C$, its projection on the image plane and the optical center are collinear, a unit vector towards $\mathbf{P}_i^C$ can be written as:

$$\tilde{\mathbf{v}}_i^C = \frac{\mathbf{v}_i^C}{|\mathbf{v}_i^C|} \qquad i \in (1, \ldots, 5) \tag{2.29}$$

The position of each marker in the camera frame can now be written in terms of the ray length:

$$\mathbf{P}_i^C = \lambda_i \tilde{\mathbf{v}}_i^C \qquad i \in (1, \ldots, 5) \tag{2.30}$$

The distance between two markers $\mathbf{P}_i^C$ and $\mathbf{P}_j^C$ can now be expressed as:

$$\begin{aligned} \mathbf{P}_i^C \mathbf{P}_j^C &= \mathbf{P}_j^C - \mathbf{P}_i^C \\ &= \lambda_j \tilde{\mathbf{v}}_j^C - \lambda_i \tilde{\mathbf{v}}_i^C \end{aligned} \tag{2.31}$$

Since all the markers are coplanar, two basis vectors are required to fully describe each marker in the plane. With $\mathbf{P}_1^C$ as the origin and L_{ij} the Euclidian distance between markers i and j, the basis vectors (Figure 2.22) can be written as:

$$\begin{aligned}\tilde{\mathbf{P}}_h &= \frac{\mathbf{P}_1^C\mathbf{P}_5^C}{|\mathbf{P}_1^C\mathbf{P}_5^C|} = \frac{\mathbf{P}_5^C - \mathbf{P}_1^C}{|\mathbf{P}_5^C - \mathbf{P}_1^C|} \\ &= \frac{\lambda_5\tilde{\mathbf{v}}_5^C - \lambda_1\tilde{\mathbf{v}}_1^C}{L_{15}} \end{aligned} \tag{2.32}$$

$$\begin{aligned}\tilde{\mathbf{P}}_v &= \frac{\mathbf{P}_1^C\mathbf{P}_2^C}{|\mathbf{P}_1^C\mathbf{P}_2^C|} = \frac{\mathbf{P}_2^C - \mathbf{P}_1^C}{|\mathbf{P}_2^C - \mathbf{P}_1^C|} \\ &= \frac{\lambda_2\tilde{\mathbf{v}}_2^C - \lambda_1\tilde{\mathbf{v}}_1^C}{L_{12}} \end{aligned} \tag{2.33}$$

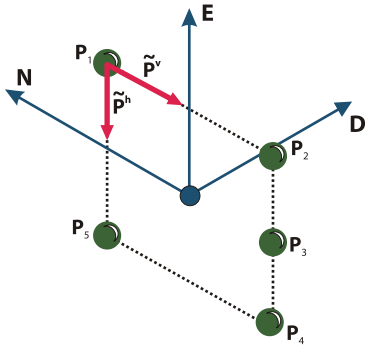


Figure 2.22: Basis vector definition

The remaining two points can now be written as a linear combination of the two basis vectors:

$$\mathbf{P}_1^C\mathbf{P}_3^C = K_1\tilde{\mathbf{P}}_h + K_2\tilde{\mathbf{P}}_v \tag{2.34}$$

$$\mathbf{P}_1^C\mathbf{P}_4^C = K_3\tilde{\mathbf{P}}_h + K_4\tilde{\mathbf{P}}_v \tag{2.35}$$

From Equation 2.34:

$$\begin{aligned}\mathbf{P}_1^C\mathbf{P}_3^C &= K_1\tilde{\mathbf{P}}_h + K_2\tilde{\mathbf{P}}_v \\ (\lambda_3\tilde{\mathbf{v}}_3^C - \lambda_1\tilde{\mathbf{v}}_1^C) &= K_1(\lambda_2\tilde{\mathbf{v}}_2^C - \lambda_1\tilde{\mathbf{v}}_1^C) + K_2(\lambda_5\tilde{\mathbf{v}}_5^C - \lambda_1\tilde{\mathbf{v}}_1^C) \\ K_1\lambda_2\tilde{\mathbf{v}}_2^C + K_2\lambda_5\tilde{\mathbf{v}}_5^C - \lambda_3\tilde{\mathbf{v}}_3^C &= (K_1 + K_2 - 1)\lambda_1\tilde{\mathbf{v}}_1^C \\ \begin{bmatrix} K_1v_{2x} & -v_{3x} & K_2v_{5x} \\ K_1v_{2y} & -v_{3y} & K_2v_{5y} \\ K_1v_{2z} & -v_{3z} & K_2v_{5z} \end{bmatrix} \cdot \begin{bmatrix} \lambda_2 \\ \lambda_3 \\ \lambda_5 \end{bmatrix} &= \begin{bmatrix} (K_1 + K_2 - 1)v_{1x} \\ (K_1 + K_2 - 1)v_{1y} \\ (K_1 + K_2 - 1)v_{1z} \end{bmatrix} \cdot \lambda_1 \end{aligned} \tag{2.36}$$

Similarly, from Equation 2.35:

$$\begin{bmatrix} K_3v_{2x} & -v_{4x} & K_4v_{5x} \\ K_3v_{2y} & -v_{4y} & K_4v_{5y} \\ K_3v_{2z} & -v_{4z} & K_4v_{5z} \end{bmatrix} \cdot \begin{bmatrix} \lambda_2 \\ \lambda_4 \\ \lambda_5 \end{bmatrix} = \begin{bmatrix} (K_3 + K_4 - 1)v_{1x} \\ (K_3 + K_4 - 1)v_{1y} \\ (K_3 + K_4 - 1)v_{1z} \end{bmatrix} \cdot \lambda_1 \tag{2.37}$$

Equations 2.36 and 2.37 represent a system of six equations and fice unknowns. Usually such an overdetermined system is solved using numerical methods, but an elegant analytical method does exist for this specific case [25].

A solution to this system can be written as:

$$\lambda_2 = \left(\frac{(K_1 + K_2 - 1)\Delta_{153}}{K_1\Delta_{253}}\right)\lambda_1 \tag{2.38}$$

$$\lambda_3 = \left(\frac{(K_1 + K_2 - 1)\Delta_{152}}{\Delta_{253}}\right)\lambda_1 \tag{2.39}$$

$$\lambda_4 = \left(\frac{(K_3 + K_4 - 1)\Delta_{152}}{\Delta_{254}}\right)\lambda_1 \tag{2.40}$$

$$\lambda_5 = \left(\frac{(K_1 + K_2 - 1)\Delta_{132}}{K_2\Delta_{253}}\right)\lambda_1 \tag{2.41}$$

with

$$\Delta_{\text{ijk}} = \begin{vmatrix} v_{ix} & v_{jx} & v_{kx} \\ v_{iy} & v_{jy} & v_{ky} \\ v_{iz} & v_{jz} & v_{kz} \end{vmatrix} \tag{2.42}$$

An expression for λ_1 can be derived in terms of the Euclidian distance between $\mathbf{P}_1^C$ and any other marker. The best results are obtained when the longest length is used for this step [25] which is the diagonal length between $\mathbf{P}_1^C$ and $\mathbf{P}_4^C$ for the chosen configuration.

$$\begin{aligned} L_{14}^2 &= \left|\mathbf{P}_1^C\mathbf{P}_4^C\right|^2 \\ &= \left|\lambda_4\tilde{\mathbf{v}}_4^C - \lambda_1\tilde{\mathbf{v}}_1^C\right|^2 \\ &= (\lambda_4 v_{4x} - \lambda_1 v_{1x})^2 + (\lambda_4 v_{4y} - \lambda_1 v_{1y})^2 + (\lambda_4 v_{4z} - \lambda_1 v_{1z})^2 \end{aligned} \tag{2.43}$$

Substituting Equation 2.40 into Equation 2.44:

$$\begin{aligned} L_{14}^2 &= \lambda_1^2\left[(K_x v_{4x} - v_{1x})^2 + (K_x v_{4y} - v_{1y})^2 + (K_x v_{4z} - v_{1z})^2\right] \\ \Rightarrow \lambda_1 &= \frac{L_{14}}{\sqrt{(K_x v_{4x} - v_{1x})^2 + (K_x v_{4y} - v_{1y})^2 + (K_x v_{4z} - v_{1z})^2}} \end{aligned} \tag{2.44}$$

with

$$K_x = \left(\frac{(K_3 + K_4 - 1)\Delta_{152}}{\Delta_{254}}\right) \tag{2.45}$$

Using the expression obtained for λ_1 in Equations 2.44 and 2.45, the expressions for λ_2

to λ_5 is given by:

$$\lambda_2 = \frac{(K_1+K_2-1)\Delta_{153}}{K_1\Delta_{253}}\left(\frac{L_{14}}{\sqrt{(K_x v_{4x}\text{-}v_{1x})^2+(K_x v_{4y}\text{-}v_{1y})^2+(K_x v_{4z}\text{-}v_{1z})^2}}\right) \quad (2.46)$$

$$\lambda_3 = \frac{(K_1+K_2-1)\Delta_{152}}{\Delta_{253}}\left(\frac{L_{14}}{\sqrt{(K_x v_{4x}\text{-}v_{1x})^2+(K_x v_{4y}\text{-}v_{1y})^2+(K_x v_{4z}\text{-}v_{1z})^2}}\right) \quad (2.47)$$

$$\lambda_4 = \frac{(K_3+K_4-1)\Delta_{152}}{\Delta_{254}}\left(\frac{L_{14}}{\sqrt{(K_x v_{4x}\text{-}v_{1x})^2+(K_x v_{4y}\text{-}v_{1y})^2+(K_x v_{4z}\text{-}v_{1z})^2}}\right) \quad (2.48)$$

$$\lambda_5 = \frac{(K_1+K_2-1)\Delta_{132}}{K_2\Delta_{253}}\left(\frac{L_{14}}{\sqrt{(K_x v_{4x}\text{-}v_{1x})^2+(K_x v_{4y}\text{-}v_{1y})^2+(K_x v_{4z}\text{-}v_{1z})^2}}\right) \quad (2.49)$$

Once the ray lengths are calculated, Equation 2.30 is used to determine the coordinates of the markers in the camera frame ($\mathbf{P}_1^C$ - $\mathbf{P}_5^C$).

Frame Transformation

There are now two sets of vectors ($\{\mathbf{P}_i^C\}$ and $\{\mathbf{P}_i^I\}$, $i = 1,\ldots,5$), coordinated in different reference frames, describing the same set of markers ($\mathbf{P}_1 - \mathbf{P}_5$). The two reference frames (camera frame and inertial frame) are related by a rotation and a translation:

$$\begin{aligned}\mathbf{P}_i^C &= \mathbf{R}\left(\mathbf{P}_i^I - \mathbf{T}\right) \\ &= \mathbf{R}\mathbf{P}_i^I - \underbrace{\mathbf{T}_c}_{\mathbf{RT}}\end{aligned} \quad (2.50)$$

where $\mathbf{R}$ is a 3×3 rotation matrix and T is a translation vector. The objective of the frame transformation part of the pose estimation algorithm is to find $\widehat{\mathbf{R}}$ and $\widehat{\mathbf{T}_c}$ to minimise

$$J = \sum_{i=1}^{5}\left|\mathbf{P}_i^C - (\mathbf{R}\mathbf{P}_i^I + \mathbf{T}_c)\right| \quad (2.51)$$

The least squares problem of Equation 2.51 can be transformed to a simpler, two-step problem where the rotation would be solved first followed by the solution of the translation [8]. The transformed cost function is given by:

$$J' = \sum_{i=1}^{5}\left|\mathbf{q}_i^C - \mathbf{R}\mathbf{q}_i^I\right| \quad (2.52)$$

with

$$\mathbf{q}_i^W \triangleq \mathbf{p}_i^I - \frac{1}{5}\sum_{j=1}^{5}\mathbf{p}_j^I \;\; \forall i \in (1,\ldots,5) \quad (2.53)$$

$$\mathbf{q}_i^C \triangleq \mathbf{p}_i^C - \frac{1}{5}\sum_{j=1}^{5}\mathbf{p}_j^C \;\; \forall i \in (1,\ldots,5) \quad (2.54)$$

The first step is to find $\widehat{\mathbf{R}}$ that minimises Equation 2.52. The second step is to determine $\widehat{\mathbf{T}}_\mathbf{c}$ and $\widehat{\mathbf{T}}$ through:

$$\begin{aligned} \widehat{\mathbf{T}}_\mathbf{c} &= \frac{1}{5}\sum_{j=1}^{5} \mathbf{p}_j^C - \widehat{\mathbf{R}}\left(\frac{1}{5}\sum_{j=1}^{5} \mathbf{p}_j^I\right) \\ \Rightarrow \widehat{\mathbf{T}} &= \widehat{\mathbf{R}}^{-1}\widehat{\mathbf{T}}_\mathbf{c} \end{aligned} \tag{2.55}$$

An algorithm presented by [8] is used to find $\widehat{\mathbf{R}}$ (Equation 2.52):

1. Calculate $\{\mathbf{q}_i^I\}$ and $\{\mathbf{q}_i^C\}$ using Equations 2.54 and 2.53.
2. Calculate the 3×3 matrix

$$\mathbf{H} \triangleq \sum_{i=1}^{5} \mathbf{q}_i^I (\mathbf{q}_i^C)^T \tag{2.56}$$

3. Find the SVD of H

$$\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \tag{2.57}$$

4. Calculate

$$\mathbf{X} = \mathbf{V}\mathbf{U}^T \tag{2.58}$$

5. Calculate the determinant of $\mathbf{X}$ to determine whether $\mathbf{X}$ is a valid rotation matrix. If $\det(X) = 1$, the rotation matrix is valid and $\widehat{\mathbf{R}} = \mathbf{X}$. If, however, $\det(X) = -1$ the algorithm has not produced the valid result. This does not necessarily present a problem since, for the coplanar case where the points are non-collinear, there is a rotation as well as reflection which could both minimise Equation 2.52. Therefore, the SVD algorithm could give either. If the determinant is -1, the result is a reflection and the rotation can be obtained by calculating:

$$\begin{aligned} \widehat{\mathbf{R}} &= \mathbf{X}' \\ &= \mathbf{V}'\mathbf{U}^T \end{aligned} \tag{2.59}$$

with

$$\mathbf{V}' = \begin{bmatrix} v_1 & v_2 & -v_3 \end{bmatrix} \tag{2.60}$$

The result of this algorithm is the rotation matrix. The translation vector can be determined by Equation 2.55 to conclude the pose estimation algorithm.

2.6.3 Comparative Study

In this section the non-iterative method is compared to the optimisation-based methods.

Runtime

In this simulation the total execution time of each of the algorithms were investigated. The same data set of simulated measurements was used for all the algorithms. The initial state vector for the iterative algorithm was chosen as the true state vector offset by 3 m in position and five degrees in attitude. The simulation for each algorithm was run for 1000 samples after which the execution time per sample was calculated. The results are shown in Figure 2.23a.

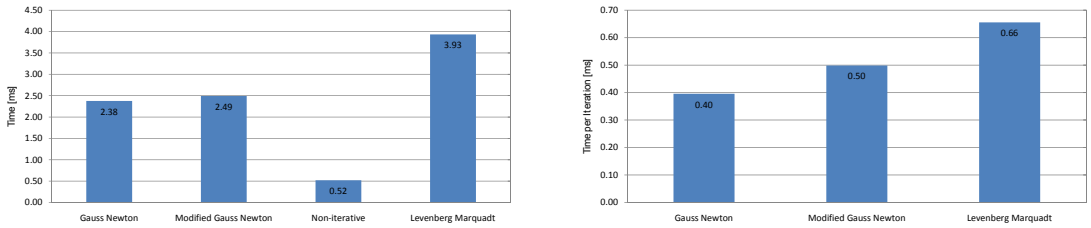
The Levenberg Marquardt method performed the worst in terms of running time while the non-iterative method was the fastest. The classic Gauss-Newton algorithm was second fastest while the modified Gauss-Newton algorithm performed worse than the classic algorithm. The classic algorithm performed the estimation in six iterations, while the modified algorithm only required five. The reason the modified algorithm executed more slowly was due to the overhead associated with the line search. This result is not indicative of the general behaviour of line search algorithms, just when the initial value is very close to the solution.

In the case of the Levenberg Marquadt algorithm, the overhead associated with the calculation of the damping parameter caused the algorithm to perform much slower than the Gauss-Newton method on which it is based. The damping parameter, however, was introduced to improve convergence when the initial value was far from the desired solution. When the current iterate is close to the solution, the step direction calculated by the Levenberg Marquadt method approximates the direction that would have been calculated by the Gauss-Newton method in the same situation. Since the estimate provided by the state estimator as an initial value is close to the solution, the Levenberg Marquadt method is essentially performing like the modified Gauss Newton method but with more overhead to update the damping parameter.

The runtime required per iteration is shown in Figure 2.23b. It is clear that the simplest algorithm (classic Gauss-Newton) performs the best for this specific cost function and initial values. The Levenberg Marquadt algorithm provides the mechanism to converge faster when the iterate is far from the solution. Similarly, the line search algorithm provides the mechanism to take larger steps in the determined direction which leads to fewer iterations. If the initial value is far from the solution, this will be a great advantage. Since the Gauss-Newton does not have guaranteed convergence it cannot be used in a system where reliability is paramount.

Parameter Sensitivity

In this simulation the sensitivity towards changes in focal length for both algorithm types was investigated. In the simulation the camera was subjected to a 2 m sinusoidal motion in the north, east and down direction at 0.4 Hz and different phases. The downwards motion was biased at 8 m above the target. The attitude was kept at constant angles of 9° for roll, 9° for pitch and 18° for yaw. Since the objective of the simulation was to test

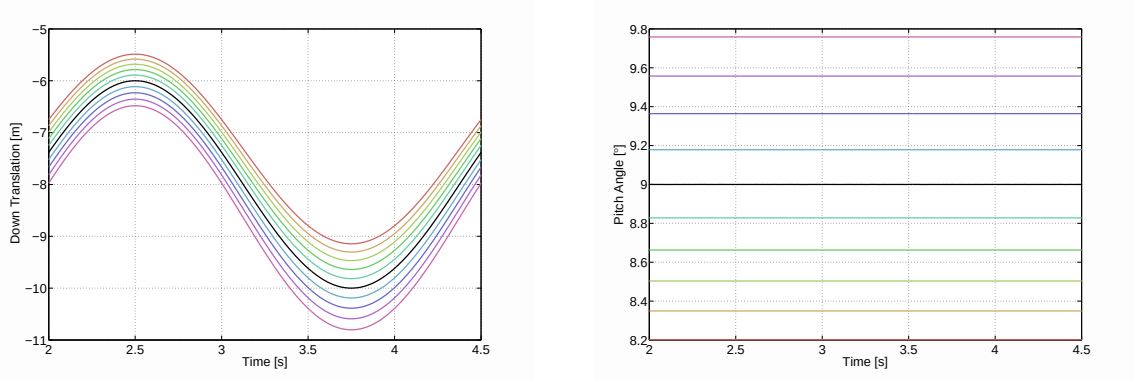


(a) Runtime comparison of all the algorithms (b) Iteration time comparison of the iterative algorithms

Figure 2.23: Runtime comparison of the pose estimation algorithms

the sensitivity of the algorithm to parameter variation, no sensor noise was added to the image measurement.

It is clear from the comparison between Figure 2.25 and Figure 2.24 that the iterative algorithm (Levenberg Marquardt) is much more sensitive to parameter variation in terms of the attitude states than the non-iterative algorithm. The position deviations are very similar, with the non-iterative algorithm performing marginally better (Table 2.4). The two Gauss-Newton algorithms had the same performance as the Levenberg Marquadt method and is not shown.

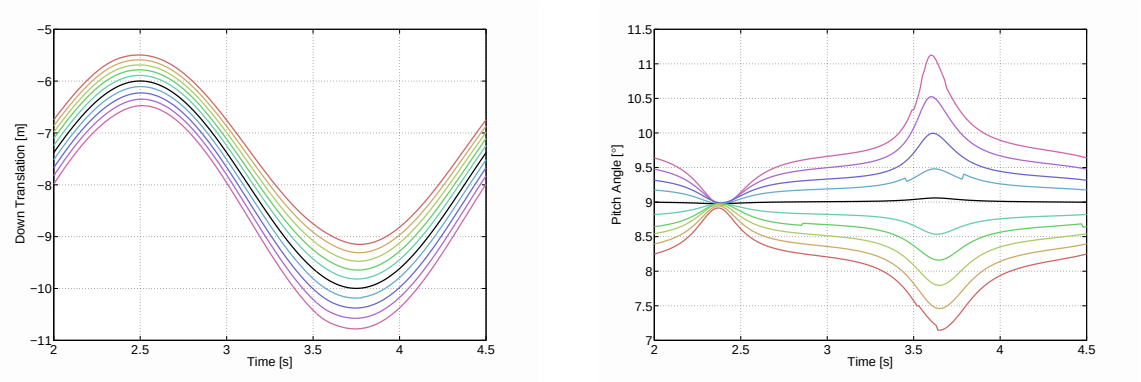


(a) Down translation (b) Pitch angle

Figure 2.24: Non-iterative algorithm: Estimated pose with 10% positive and negative adjustment of the focal length.

Table 2.4: Comparison of the maximum state deviation due to parameter change

Method	ϕ [°]	θ [°]	ψ [°]	N [m]	E [m]	D [m]
Iterative (Levenberg Marquardt)	1.2457	1.9121	0.2555	0.5341	0.1620	0.8145
Non-Iterative	0.5420	0.7989	0.0853	0.3183	0.0918	0.8033



(a) Down translation (b) Pitch angle

Figure 2.25: Iterative algorithm: Estimated pose with 10% positive and negative adjustment of the focal length.

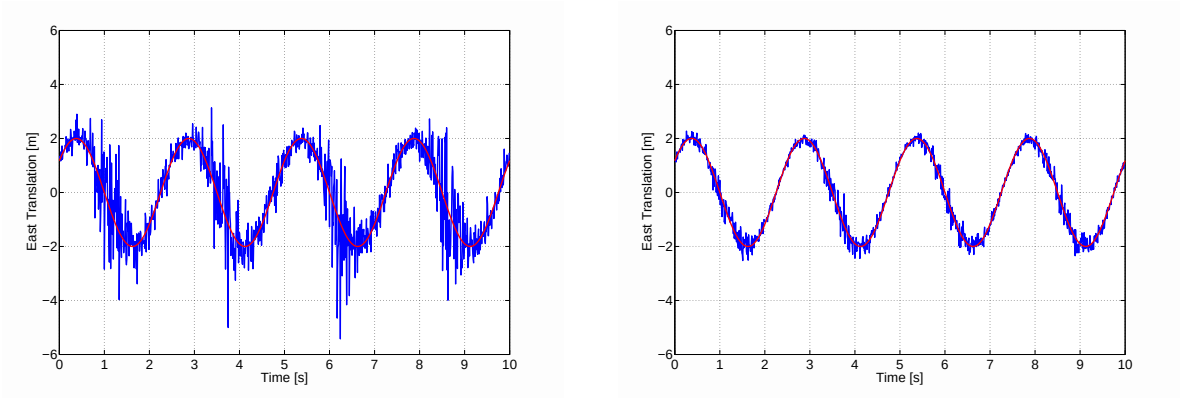
Noise Robustness

The aim of this simulation was to determine the effect of noise on the algorithm. The noise in a vision system normally manifests as pixels not having the correct intensity due to charge leakage between adjacent pixels [92] and shot noise typical to semiconductor devices [58]. Since the average of the centroids are used in the algorithms, individual noisy pixels do not directly influence the system. Instead, these pixels collectively influence the average. As a result, the effect of a small number of noise pixels is hardly noticeable. The ability of the noise pixels to influence the centroid is very dependent on the mass of the region. For a large mass, stray pixels will not have a large influence on the centroid. On the contrary, if the mass is small, the stray pixels have a more noticeable effect on the centroid. Consequently, accuracy increases with decreasing range, which is ideal for a sensor used during a landing procedure.

In the simulation, the centroids in the generated image were corrupted with a single pixel variance noise source. The East translation estimates are shown in Figure 2.26. The variances for the full state vector obtained with both methods are shown in Table 2.5. The two Gauss-Newton algorithms had the same performance as the Levenberg Marquadt method and is not shown. It is clear that the non-iterative method is slightly more robust than the iterative method.

Table 2.5: Comparison of the maximum state deviation due to noise

Method	ϕ [°]	θ [°]	ψ [°]	N [m]	E [m]	D [m]
Iterative (Levenberg Marquadt)	0.0063	0.0064	0.0002	2.6696	2.5123	2.0429
Non-Iterative	0.0002	0.0004	0.0000	2.0659	2.0298	1.9784



(a) Levenberg Marquadt

(b) Non-iterative

Figure 2.26: Comparison of the effect of noise on the pose estimation algorithms

Conclusion

From the simulations in the previous sections, it is clear that the non-iterative method is superior in terms of runtime, parameter sensitivity and noise robustness.

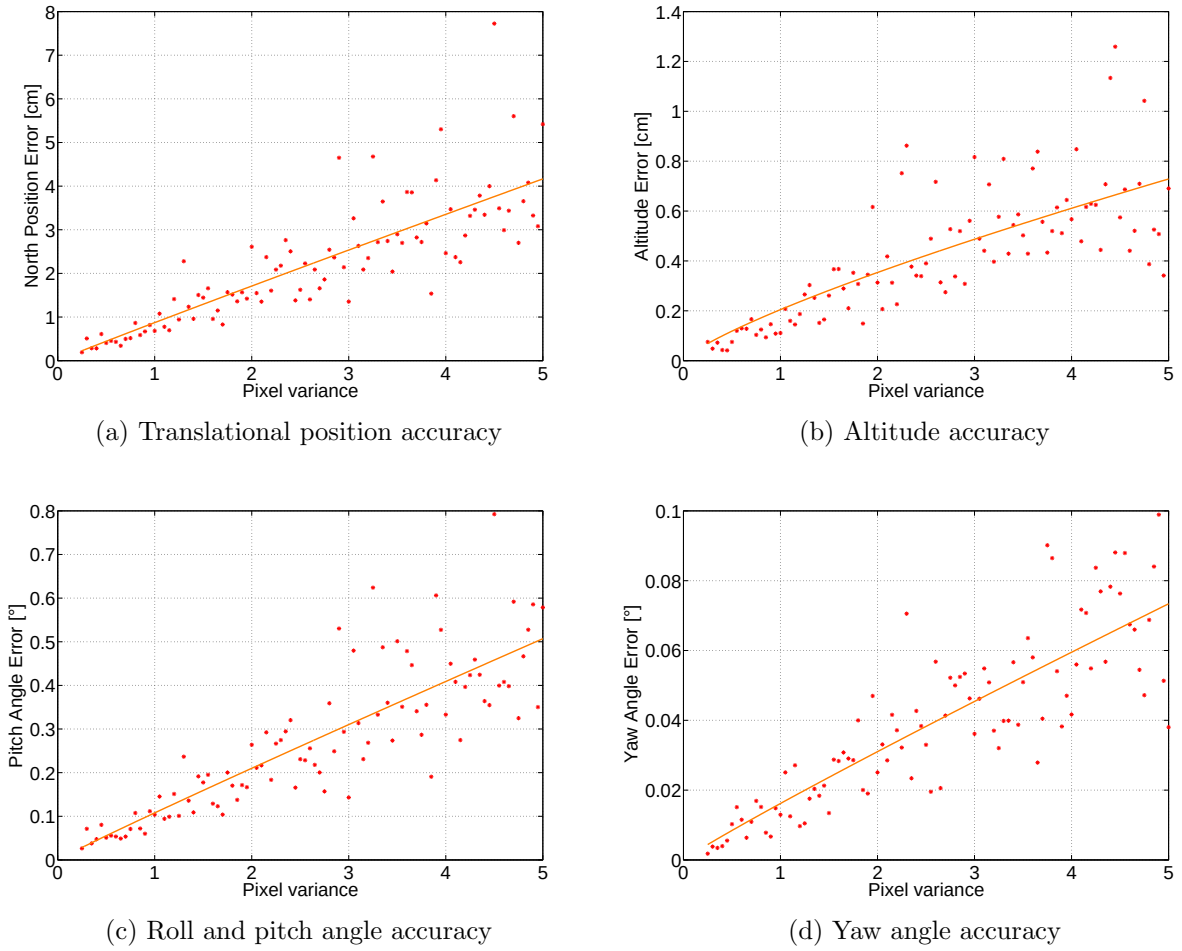
2.7 Accuracy

The aim of the VPAM system is to sufficiently improve state estimates in order to perform autonomous landings within a circular region with a 1 m radius. It is important to investigate the accuracy of the algorithms developed in this chapter to determine (through simulation in Chapter 6) that the required level of accuracy is achievable. Possible noise sources that can effect the accuracy of the system includes image distortion, image noise and quantisation.

Radial distortion caused by the optics causes the centroids to move in the image plane. This distortion must be reversed before the centroids can be used for measurement applications. Unfortunately these algorithms are not perfect and can result in the centroids being slightly offset from their true position [28]. Typical pixel errors after distortion correction are very small: 0.1 pixels [26], 1.3 pixels [28] and 0.001 pixels [84] as typical examples. Radial distortion and the required compensation is discussed in Section 7.2.3.

Image noise is the direct effect of noise in the logic structures of the image sensor and affect the intensity of random pixels in the image sensor. In the practical system, the centroid is determined through the average of the high intensity regions in the image (Section 5.2.1) which dilutes the effect a single noise-affected pixel has on the centroid's position. With smaller high-intensity regions at higher altitudes, noise affected pixels can have a larger effect on the position, but shifts larger than a few pixels are not expected. Since a practical image sensor consist of a number of discrete pixels, small quantisation errors are made when the real image plane is mapped onto discrete pixels. The worst case effect of the quantisation can be represented by a single pixel shift.

From the above discussion, the main sources of noise each contribute, in the worst case, a few pixels to the overall shift of a centroid. Since it is difficult to quantify the error, a simulation was done where the image centroids were shifted by distances equivalent to a small number of pixels to determine the effect it has on the pose estimation accuracy. The results for translational position accuracy are shown in Figure 2.27a, altitude in Figure 2.27b, roll and pitch angle in Figure 2.27c and yaw in Figure 2.27d. This simulation was performed at an altitude of 4 m.



(a) Translational position accuracy

(b) Altitude accuracy

(c) Roll and pitch angle accuracy

(d) Yaw angle accuracy

Figure 2.27: Effect of noise on pose estimation accuracy

It is typical for VPAM systems to have accuracy in the order of 5 cm or below [33, 55, 23]. As a result, noise with a standard deviation of 3 pixels was chosen in Figure 2.27 to use as an approximation of the measurement noise required in the remainder of the system development and simulation. The chosen measurement noise is given in Table 2.6:

2.8 Summary

In this chapter the algorithms required to determine the position and attitude of a camera, given an image of landing target markers and *a priori* geometric knowledge of the landing

Table 2.6: Approximate measurement noise for the VPAM system

Quantity	**1σ Deviation**
Translational position [cm]	2.5
Altitude [cm]	0.5
Roll and pitch angle [°]	0.3
Yaw angle [°]	0.05

target, were derived. The correspondence matching algorithm is reponsible for finding the set of projections in the image which can be identified as the corresponding target markers in world coordinates. The algorithm was derived in Section 2.5.2 and an emperical threshold that determines the functionality and robustness of the algorithm was developed in Section 2.5.3. The functionality was confirmed in Section 2.5.4 through a series of simulations testing the algorithm's ability to handle occlusions and the introduction of false markers.

Two methods of pose estimation were introduced in Section 2.6: an iterative, optimisation-based method and a non-iterative vector-based method. An optimisation-based method was developed in Section 2.6.1 and a number of numerical methods that could be used for solving the problem were evaluated. In Section 2.6.2 the development of a non-interative pose estimation method was outlined. The two methods were compared in Section 2.6.3 in terms of runtime, robustness against noise and parameter variation and the non-iterative method proved superior in all three cases.

The expected accuracy of the system was estimated through simulation and compared to similar methods in the literature.

Chapter 3

Visual Augmented State Estimator

3.1 Introduction

A state estimator is used to obtain the translational and rotational states of a vehicle, even if they are not directly measureable. This is achieved by combining the measurements from a number of sensors and making use of the kinematic relationships between the states and measurements. A full state kinematic estimator was designed in the ESL [37]. In this estimator the measurements from strapped down gyroscopes are used to calculate the platform's attitude. Using the attitude, the strapped-down accelerometer measurements can be coordinated in an inertial reference frame. After compensating for gravitational acceleration, the measurements are integrated twice to determine the platform's inertial frame coordinated velocity and position.

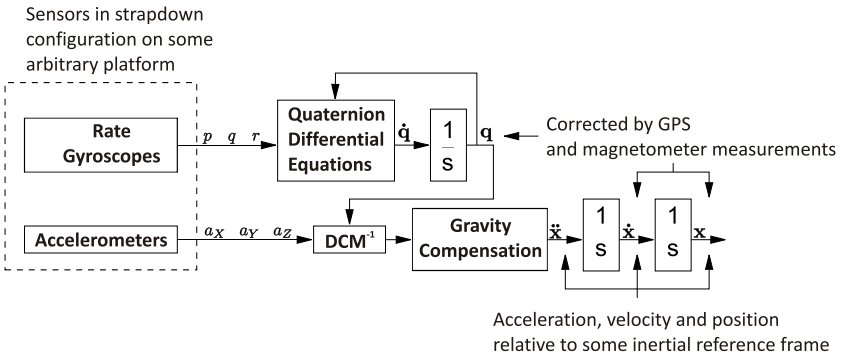


Figure 3.1: The original kinematic state estimator structure [37]

Due to the approximated integration process and biases on the inertial sensors, the integrated states (position, velocity, attitude) will deteriorate over time. The deterioration can be kept under control by using direct and indirect measurements of the states provided by the GPS and magnetometer to update the integrated states. This structure is shown in Figure 3.1.

This estimator, combined with a set of low-cost sensors, cannot provide sufficiently accurate measurements to faciliate autonomous landings and the following modifications were made to obtain the required level of accuracy [92]:

1. Fusion of delayed, variable frequency, visual measurements.
2. Using a linearised flat-earth model for position instead of latitude and longitude.
3. Using Euler angles instead of quaternions since extreme attitudes were not part of the flight envelope.
4. Dividing the estimator into two separate, lower order estimators. The position and velocity estimator was implemented using a linear Kalman filter (KF) to estimate the position and velocity states $[V_N\ V_E\ V_D\ P_N\ P_E\ P_D]$. The attitude estimator was implemented using a non-linear Kalman filter (EKF) to estimate the attitude states $[\phi\ \theta\ \psi]$.

The high velocities of the platform used in [92] made it necessary to compensate for the measurement delays in the GPS (310 ms) and vision systems (variable between 100 ms and 800 ms). This was achieved by maintaining a circular list of all measurements for a two second period and when a GPS or vision system measurement is received stepping back the required period of time, doing a measurement update and re-propagating to the current point in time. This is a relatively costly method in terms of storage and computational efficiency[1].

Due to the relatively slow velocities of a helicopter using conventional hover based control systems [69], especially during the landing phase, the measurement delays can be ignored without a significant decrease in accuracy. This simplified estimator has been developed and tested during a number of engineering projects at the ESL but was not documented and analysed. This chapter outlines the derivation of this simplified version as well as the augmentation of visual measurements.

The structure of the conventional estimator and the modification for visual measurements is overviewed in Section 3.2. The derivation of the position and velocity estimator is shown in Section 3.3 and the derivation of the attitude estimator is shown in Section 3.4. To avoid transitional effects when the vision based measurements become valid or invalid, a smooth transition scheme is detailed in Section 3.6. Simulation results are presented in Section 3.7 and and concluding remarks are shown in Section 3.8.

3.2 Estimator Structure

The same divided approach introduced by [92] is used since the separation of attitude and translational dynamics is intuitive and simplifies debugging and analysis. The structure is shown in Figure 3.2.

[1]The implementation of [92] was done on a comparitively powerful Celeron 300 MHz.

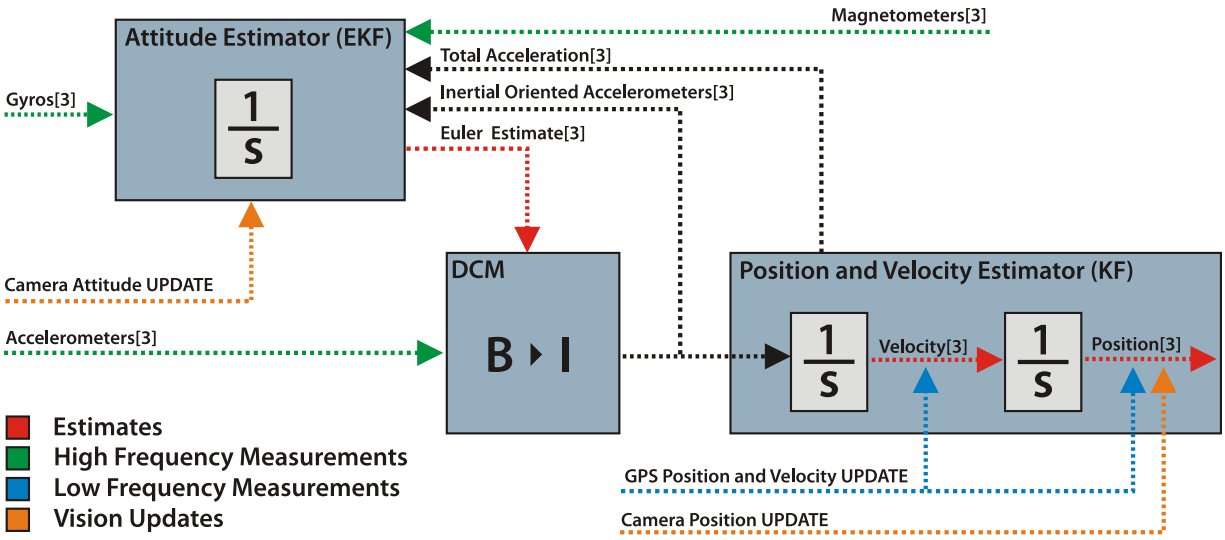


Figure 3.2: The divided simplified estimator structure

The first estimator maintains the attitude of the airframe in the inertial reference frame. Since the attitude dynamics are inherently non-linear, an extended Kalman filter (EKF) must be used to estimate the attitude state vector $[\phi\ \theta\ \psi]$. This second estimator maintains the translational position and velocity of the airframe in an inertial frame. These dynamics are linear and encapsulates the kinematic relationship between position and velocity. As a result a conventional Kalman filter (KF) will be sufficient to estimate the position and velocity state vector $[V_N\ V_E\ V_D\ P_N\ P_E\ P_D]$.

3.3 Position and Velocity Estimator

The development of the position and velocity estimator is outlined in this section. It involves the derivation of the kinematic relationships in Section 3.3.1 and filter equations in Section 3.3.2.

3.3.1 Linear Dynamics

Before the Kalman filter is developed, the translational dynamics must be determined. With the knowledge that the output of the attitude estimator will be inertially oriented accelerometer measurements the linear translational dynamics can be represented in the continuous domain by Equation 3.1:

$$\begin{bmatrix} \dot{P}_N \\ \dot{P}_E \\ \dot{P}_D \\ \dot{V}_N \\ \dot{V}_E \\ \dot{V}_D \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}_{pv}} \cdot \begin{bmatrix} P_N \\ P_E \\ P_D \\ V_N \\ V_E \\ V_D \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{B}_{pv}} \cdot \underbrace{\begin{bmatrix} a_N \\ a_E \\ a_D + g \end{bmatrix}}_{\mathbf{u}_{pv}} + \mathbf{w}_{pv} \tag{3.1}$$

where $\mathbf{w}_{pv}$ is the process noise associated with the plant. The stochastic properties of the process noise is described by the process noise covariance matrix $\mathbf{Q}_{pv}$.

The continuous linear dynamics can be represented in the discrete time domain by Equation 3.2:

$$\begin{bmatrix} P_N \\ P_E \\ P_D \\ V_N \\ V_E \\ V_D \end{bmatrix}_{k+1} = \mathbf{\Phi}_{pv}(k) \cdot \begin{bmatrix} P_N \\ P_E \\ P_D \\ V_N \\ V_E \\ V_D \end{bmatrix}_{k} + \mathbf{\Gamma}_{pv}(k) \cdot \begin{bmatrix} a_N \\ a_E \\ a_D + g \end{bmatrix} + \mathbf{w}_{pv}(k) \tag{3.2}$$

using the approximations [27] in Equations 3.3 and 3.4:

$$\begin{aligned} \mathbf{\Phi} = \mathbf{I} + \mathbf{A} \cdot \Delta\mathrm{T} + \frac{\mathbf{A}^2 \cdot \Delta\mathrm{T}^2}{2!} + \ldots &\approx \mathbf{I} + \mathbf{A} \cdot \Delta\mathrm{T} \\ \Rightarrow \mathbf{\Phi}_{pv}(k) &\approx \mathbf{I} + \mathbf{A}_{pv} \cdot \Delta\mathrm{T} \\ &= \begin{bmatrix} 1 & 0 & 0 & \Delta\mathrm{T} & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta\mathrm{T} & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta\mathrm{T} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.3}$$

and

$$\begin{aligned} \mathbf{\Gamma} = \mathbf{B} \cdot \Delta\mathrm{T} + \mathbf{B} \cdot \frac{\mathbf{A} \cdot \Delta\mathrm{T}^2}{2!} + \ldots &\approx \mathbf{B} \cdot \Delta\mathrm{T} \\ \Rightarrow \mathbf{\Gamma}_{pv}(k) &\approx \mathbf{B_{pv}} \cdot \Delta\mathrm{T} \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \Delta\mathrm{T} & 0 & 0 \\ 0 & \Delta\mathrm{T} & 0 \\ 0 & 0 & \Delta\mathrm{T} \end{bmatrix} \end{aligned} \tag{3.4}$$

These approximations remain valid as long as the discrete sampling time remains much shorter than the system time constants [27].

Since the state estimates in this project are updated using two sensors (GPS module and VPAM node), there are two distinct output matrices: one for GPS updates and another for vision updates. The GPS updates both the translational position and velocity:

$$\mathbf{y}_{pv,GPS}(k) = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{H}_{pv,GPS}} \cdot \underbrace{\begin{bmatrix} P_N \\ P_E \\ P_D \\ V_N \\ V_E \\ V_D \end{bmatrix}}_{\widehat{\mathbf{x}}_{pv}^{-}(k)} + \mathbf{v}_{pv,GPS}(k) \tag{3.5}$$

where $\mathbf{v}_{pv,GPS}$ is the measurement noise associated with the GPS module. The noise's stochastic properties are described by the measurement noise covariance matrix $\mathbf{R}_{pv,GPS}$.

The camera, however, only updates the translational position:

$$\mathbf{y}_{pv,Cam}(k) = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{H}_{pv,Cam}} \cdot \underbrace{\begin{bmatrix} P_N \\ P_E \\ P_D \\ V_N \\ V_E \\ V_D \end{bmatrix}}_{\widehat{\mathbf{x}}_{pv}^{-}(k)} + \mathbf{v}_{pv,Cam}(k) \tag{3.6}$$

where $\mathbf{v}_{pv,Cam}$ is the measurement noise associated with the VPAM node. The stochastic properties is described by the measurement noise covariance matrix $\mathbf{R}_{pv,Cam}$.

Through observability tests [27] it can be shown that the system dynamics are observable through $\mathbf{H}_{pv,Cam}$ and $\mathbf{H}_{pv,GPS}$.

3.3.2 Filter Equations

The Kalman filter algorithm is governed by the following set of equations [37, 27]:

1. The previous best state is propagated forward by one sample period using the plant dynamics and the deterministic input:

$$\widehat{\mathbf{x}}_{pv}^{-}(k+1) = \mathbf{\Phi}\widehat{\mathbf{x}}_{pv}(k) + \mathbf{\Gamma}\mathbf{u}_{pv}(k) \tag{3.7}$$

   The error covariance of the the propagated state is calculated through:

$$\mathbf{P}_{pv}^{-}(k+1) = \mathbf{\Phi}\mathbf{P}_{pv}(k)\mathbf{\Phi}^T + \mathbf{Q}_{pv}(k) \tag{3.8}$$

2. When a measurement update is available, the filter gains can be calculated using Equations 3.5, 3.8 and 3.6.

$$\mathbf{L}_{pv,i}(k+1) = \mathbf{P}_{pv}^{-}(k+1)\mathbf{H}_{pv,i}{}^{T} \cdot \left[\mathbf{H}_{pv,i}\mathbf{P}_{pv}^{-}(k+1)\mathbf{H}_{pv,i}{}^{T} + \mathbf{R}_{pv,i}(k+1)\right]^{-1} \quad (3.9)$$

where $i \in [\text{GPS},\ \text{Cam}]$ and $\mathbf{R}_{pv,i}$ is the noise covariance of the applicable sensors. These gains are formulated in such a way as to minimise the errors of the best state estimate.

3. Perform the required innovation updates using the filter gains (Equation 3.9) and the error between the actual and propagated measurements:

$$\widehat{\mathbf{x}}_{pv}(k+1) = \widehat{\mathbf{x}}_{pv}^{-}(k+1) + \mathbf{L}_{pv,i}(k+1) \cdot \left[\mathbf{y}_{pv,i}(k+1) - \mathbf{H}_{pv,i}\widehat{\mathbf{x}}_{pv}^{-}(k+1)\right] \quad (3.10)$$

The error covariance of the current best state estimate can be calculated using Equations 3.9 ans 3.8:

$$\mathbf{P}_{pv}(k) = [\mathbf{I} - \mathbf{L}_{pv,i}(k)\mathbf{H}_{pv,i}] \cdot \mathbf{P}_{pv}^{-}(k) \quad (3.11)$$

$\mathbf{P}_{pv}(k)$ is a common state covariance matrix and indicates the quality of the current best state estimate.

The low frequency sensors (camera and GPS) are used to provide measurement updates to prevent deterioration due to the propagation of low accuracy accelerometers. The Kalman filter optimally fuses measurements from multiple sources to create an accurate high frequency state vector by propagating the accelerometer measurements (step 1) and performing measurement updates when new measurements become available (step 2 and 3). The degree to which the measurement updates influence the state estimates is discussed in Section 3.5.

Since the dynamics of the position and velocity estimator remain unchanged during the entire operating envelope, the Kalman gain ($\mathbf{L}_{pv,i,\infty}(k)$) can be calculated offline and considered constant during the operation of the estimator. In this project the gain is calculated using the steady state equations derived in [27]. This modification simplifies the estimator algorithm to:

1. The previous best state is propagated forward by one sample period using the plant dynamics and the deterministic input:

$$\widehat{\mathbf{x}}_{pv}^{-}(k) = \mathbf{\Phi}\widehat{\mathbf{x}}_{pv}(k-1) + \mathbf{\Gamma}\mathbf{u}_{pv}(k-1) \quad (3.12)$$

2. The required innovation update is performed using the appropriate steady-state filter gain from Equation 3.9 and the error between the actual and propagated measurements:

$$\widehat{\mathbf{x}}_{pv}(k) = \widehat{\mathbf{x}}_{pv}^{-}(k) + \mathbf{L}_{pv,i,\infty}(k) \cdot \left[\mathbf{y}_{pv,i}(k) - \mathbf{H}_{pv,i}\widehat{\mathbf{x}}_{pv}^{-}(k)\right] \quad (3.13)$$

Since the stochastic update and gain calculation involves matrix inversion and a number of multiplication operations, this simplifcation results in a significant increase in computational efficiency.

3.4 Attitude Estimator

The development of the attitude estimator is outlined in this section. It involves the derivation of the kinematic relationships in Section 3.4.1 and filter equations in Section 3.4.2.

3.4.1 Dynamics

The non-linear state and output dynamics can be represented by:

$$\begin{aligned} \dot{\mathbf{x}}_{att} &= \mathbf{f}\left(\mathbf{x}_{att}, \mathbf{u}_{att}\right) + \mathbf{w}_{att} \\ &= \underbrace{\begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \cdot \begin{bmatrix} p \\ q \\ r \end{bmatrix}}_{\mathbf{f}(\mathbf{x}_{att}, \mathbf{u}_{att})} + \mathbf{w}_{att} \end{aligned} \tag{3.14}$$

with outputs:

$$\mathbf{y}_{att} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{H}_{att}} \cdot \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} + \mathbf{v}_{att,i} \tag{3.15}$$

where $\mathbf{w}_{att}$ is the process noise associated with the attitude dynamics. The stochastic properties of the process noise are described by the the process noise covariance matrix $\mathbf{Q}_{att}$. The measurement noise is represented by $\mathbf{v}_{att,i}$ with the stochastic properties described by $\mathbf{R}_{att,i}$ where $i \in [\text{GPS},\ \text{Cam}]$.

3.4.2 Filter Equations

With the dynamics defined, the Kalman filter can be implemented with the following algorithm:

1. Since the design of the Kalman filter requires a linear plant model, a first order linearisation must be performed at the start of each time step. This is done by calculating the respective Jacobian matrices and evaluating them at $\widehat{\mathbf{x}}_{att}$ and $\widehat{\mathbf{u}}_{att}$.

$$\mathbf{A}_{att} = \left[\frac{\partial \mathbf{f}\left(\mathbf{x}_{att}, \mathbf{u}_{att}\right)}{\partial \mathbf{x}_{att}}\right]_{\widehat{\mathbf{x}}_{att}, \mathbf{u}_{att}} \tag{3.16}$$

$$\mathbf{B}_{att} = \left[\frac{\partial \mathbf{f}\left(\mathbf{x}_{att}, \mathbf{u}_{att}\right)}{\partial \mathbf{u}_{att}}\right]_{\widehat{\mathbf{x}}_{att}, \mathbf{u}_{att}} \tag{3.17}$$

2. Using the same approximations as in Equations 3.3 and 3.4, the discrete linear dynamics can be calculated:

$$\mathbf{\Phi}_{att}(k) = \begin{bmatrix} 1 + \Delta\mathrm{T} \cdot \mathrm{t}_\theta(\mathrm{c}_\phi q - \mathrm{s}_\phi r) & \Delta\mathrm{T} \cdot \sec\theta^2(\mathrm{s}_\phi q + \mathrm{c}_\phi r) & 0 \\ -\Delta\mathrm{T} \cdot (\mathrm{s}_\phi q + \mathrm{c}_\phi r) & 1 & 0 \\ \Delta\mathrm{T} \cdot \sec\theta(\mathrm{c}_\phi q - \mathrm{s}_\phi r) & \Delta\mathrm{T} \cdot \sec\theta \cdot \mathrm{t}_\theta(\mathrm{s}_\phi q + \mathrm{c}_\phi r) & 1 \end{bmatrix}_{\widehat{\mathbf{x}}_{att}, \mathbf{u}_{att}} \tag{3.18}$$

$$\mathbf{\Gamma}_{att}(k) = \Delta\mathrm{T} \cdot \begin{bmatrix} 1 & \mathrm{s}_\phi \mathrm{t}_\theta & \mathrm{c}_\phi \mathrm{t}_\theta \\ 0 & \mathrm{c}_\phi & -\mathrm{s}_\phi \\ 0 & \mathrm{s}_\phi \sec\theta & \mathrm{c}_\phi \sec\theta \end{bmatrix}_{\widehat{\mathbf{x}}_{att}} \tag{3.19}$$

where $\mathrm{s}_\alpha = \sin(\alpha)$, $\mathrm{c}_\alpha = \cos\alpha$ and $\mathrm{t}_\alpha = \tan\alpha$

3. Propagate the previous best state estimate forward by numerically integrating the non-linear dynamics over one sample instance using Euler integration:

$$\begin{aligned} \widehat{\mathbf{x}}^{-}_{att}(k) &= \widehat{\mathbf{x}}_{att}(k-1) + \dot{x}(k-1)att \cdot \Delta\mathrm{T} \\ &= \widehat{\mathbf{x}}_{att}(k-1) + \mathbf{f}(\mathbf{x}_{att}, \mathbf{u}_{att})\Delta\mathrm{T} \end{aligned} \tag{3.20}$$

The error covariance of the the propagated state is calculated using:

$$\mathbf{P}^{-}_{att}(k) = \mathbf{\Phi}_{att}(k-1)\mathbf{P}_{att}(k-1)\mathbf{\Phi}^T + \mathbf{Q}_{att}(k-1) \tag{3.21}$$

4. When a measurement becomes available, the filter gain can be calculated:

$$\mathbf{L}_{att,i}(k) = \mathbf{P}^{-}_{att}(k)\mathbf{H}_{att,i}{}^T \cdot \left[\mathbf{H}_{att,i}\mathbf{P}^{-}_{att,i}(k)\mathbf{H}_{att,i}{}^T + \mathbf{R}_{att,i}\right]^{-1} \tag{3.22}$$

where $i \in$ [GPS, Cam]. If a GPS measurement becomes available, it must be transformed to an attitude update using the TRIAD procedure outlined in [37]. The result of this algorithm will be used as the measurement during the innovation step. If a visual measurement becomes available, it can be used directly.

5. Perform the innovation update to obtain the current best state estimate.

$$\widehat{\mathbf{x}}_{att}(k) = \widehat{\mathbf{x}}^{-}_{att}(k) + \mathbf{L}_{att,i}(k) \cdot \left[\mathbf{y}_{att,i}(k) - \mathbf{H}_{att,i}\widehat{\mathbf{x}}^{-}_{pv}(k)\right] \tag{3.23}$$

6. The error covariance of the current best state estimate can be calculated using Equations 3.22 and 3.21:

$$\mathbf{P}_{att}(k) = [\mathbf{I} - \mathbf{L}_{att,i}(k)\mathbf{H}_{att,i}] \cdot \mathbf{P}^{-}_{att}(k) \tag{3.24}$$

A common covariance matrix ($\mathbf{P}_{att}$) is used for both sensors and indicates the quality of the attitude state estimate.

3.5 Measurement Updates

The Kalman gain encapsulates to what degree a measurement update affects the state estimate using the process and measurement noise. If sensor noise is less than process noise, the Kalman gain will allow the measurement update to significantly influence the state estimate. If, however, the sensor noise is more than the process noise, the Kalman gain will prevent the measurement update from having a significant effect on the state estimate. The Kalman gain does not take into account when multiple sensors are used for measurement updates. It only takes the relevant process and measurement noise for a particular sensor into account. It has no regard for the measurement noise of another sensor that is used for measurement updates in the same estimator. The result is a situation where a noisy sensor (with a Kalman gain that is significant enough to affect the state estimates) deteriorates the improvement made by a less noisy sensor. This is the case with the GPS and camera updates in this project.

When visual lock has not yet been obtained, the GPS is the only source of measurement updates. Since it is a conventional low-cost device, the position measurements are not particularly accurate ($\sigma_{GPS,Pos} = 4$ m [37]) but it does have a large enough effect on the state estimate to keep it from diverging. When vision measurements become available, the position accuracy is much better ($\sigma_{Cam,Pos} = 2.5$ cm) and a high Kalman gain is used for the update. Since GPS measurements are still received while the helicopter is within visual range, measurement updates for both the GPS and vision system occur. The vision sensor (with a large Kalman gain) will improve the state estimate, but the GPS update (with a smaller Kalman gain, but still significant enough to affect the state estimates) will slightly deteriorate the state estimate. This effect for the North position state is shown in Figure 3.3. The error between the true state and the two estimates is shown in Figure 3.4.

One solution is to increase the ratio of GPS measurement noise to process noise when visual lock has been obtained. This results in a smaller Kalman gain for GPS updates and reduces the effect it has on the state estimates. Practically, however, this is not the most efficient method, since the calculations will still be performed without having a noticable effect on the state estimates. As a result a logic procedure is used to disable GPS measurement updates when visual lock has been obtained. When visual lock is lost, the GPS measurement updates are re-enabled. The GPS module is not disabled during this period and its built-in Kalman filter is kept running to prevent delays associated with obtaining a satellite fix and the first valid solution.

3.6 Sensor Transition

Before visual lock has been obtained, the state estimate will approximate the GPS measurement. Since the GPS is relatively inaccurate compared to the VPAM system, there could be a disparity between the two measurements and, as a result, between the state estimate and the camera measurement when visual lock has been obtained. This, combined

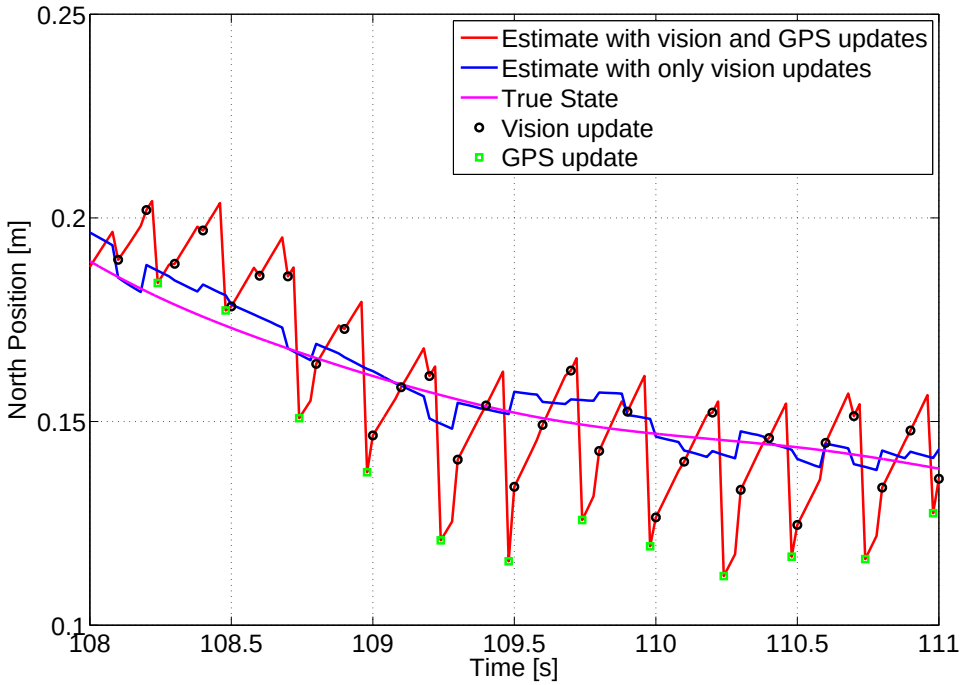


Figure 3.3: The effect of GPS measurement updates when visual lock has been obtained

with a large Kalman gain when visual lock has been obtained, will lead to a relatively large innovation step in Equation 3.13 resulting in a large step in the state estimate (Figure 3.5a). This is equivalent to applying step reference to a control system since both result in a step error which the control system will attempt to regulate. The same argument applies when visual lock is lost and the state estimate is close to the camera measurement. There could be disparity between the state estimate and the latest GPS measurement which will also result in the state estimate step. These steps will result in sudden motions of the helicopter platform which is undesirable since it can cause visual lock to be lost and re-acquired in rapid succession, causing more steps in the estimates.

The proposed solution is to have a smooth transition between sensor domains (acquiring visual lock and using only the vision system or losing visual lock and using only GPS) by gradually phasing in the new sensor measurements. The result of this transition is smooth state estimates. For the position and velocity estimator the adjustment is performed during a measurement update by modifying the error between the measurement ($\mathbf{y}_{pv,i}$) and the expected measurement ($\mathbf{H}_{pv,i}\mathbf{x}$) in the innovation step (Equation 3.13). Where i indicates the sensor used in the domain that is being transitioned to.

The transition is performed by creating an adjusted measurement $\mathbf{y}_{pv,i}{}'$ that will be used as the measurement in Equation 3.13. This adjustment (Equation 3.25) is performed every time a measurement update of the new domain takes place. During each adjustment the ageing factor λ_k (which is initialised to 1 at the domain transition) is decreased by a small amount (Equation 3.26). When it reaches 0, the transition is complete and unmodified

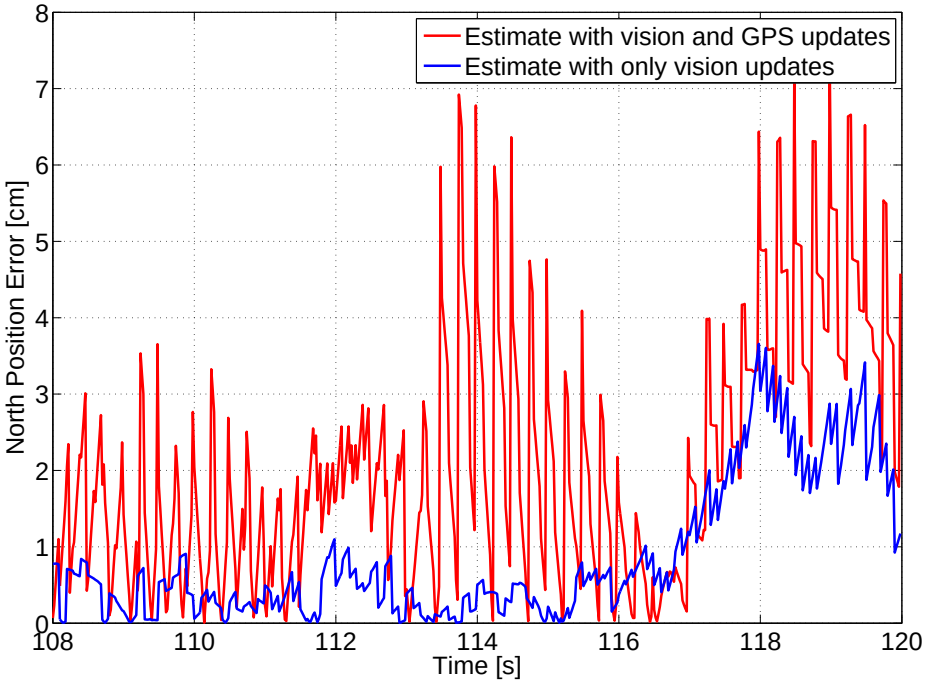


Figure 3.4: The state error when GPS measurements are used when visual measurements are available

sensor measurements are used in Equation 3.13.

$$\mathbf{y}_{pv,i}{}' = (\mathbf{H}_{pv,i}\mathbf{x} - \mathbf{y})\lambda_k + \mathbf{y} \tag{3.25}$$

$$\lambda_{k+1} = \lambda_k - \tau \tag{3.26}$$

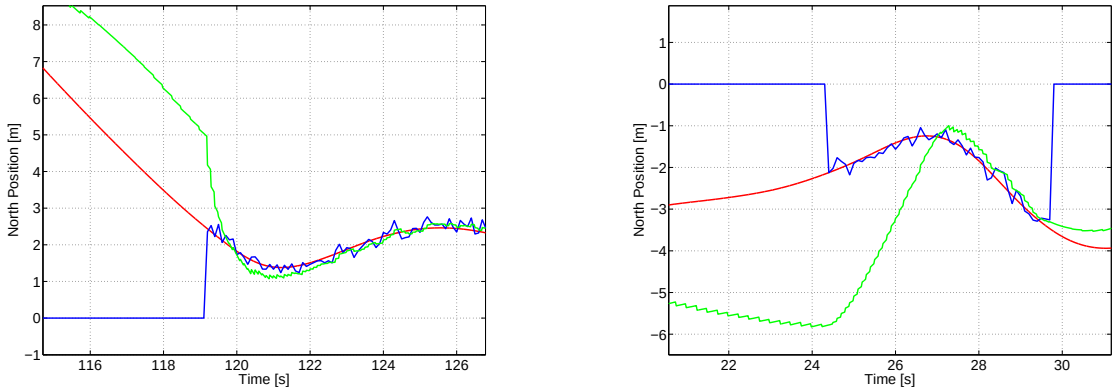
where τ determines the transition period T_p:

$$T_p = \tau \cdot f \tag{3.27}$$

and f is the update frequency of the new sensor domain. The result of the smooth transition with a transition period of 1 second is shown in Figure 3.5b for the north position state. When the visual updates become available, the measurements are adjusted for 1 second after which the period until convergence is due to the dynamics of the Kalman filter. The step in the state estimate when a smooth transition is not applied is shown in Figure 3.5a. The same method is also applied in the attitude estimator to ensure smooth state estimates at the transition of sensor domains.

3.7 Results

A simulation was set up to test the estimator in the GPS sensor domain as well as in the visual domain. The down state estimate and downwards velocity state estimate are shown



(a) No smooth transition result in estimate steps (b) Smooth transition prevents estimate steps

Figure 3.5: Effect of smooth transition on state estimates during sensor domain transitions

in Figures 3.6 and 3.7 to confirm the functionality of the estimator. The improvement of the state estimates when visual lock has been acquired is clear. The effect of the smooth transition algorithm can also be seen at the transition of sensor domains. The remaining states are shown in Appendix C.

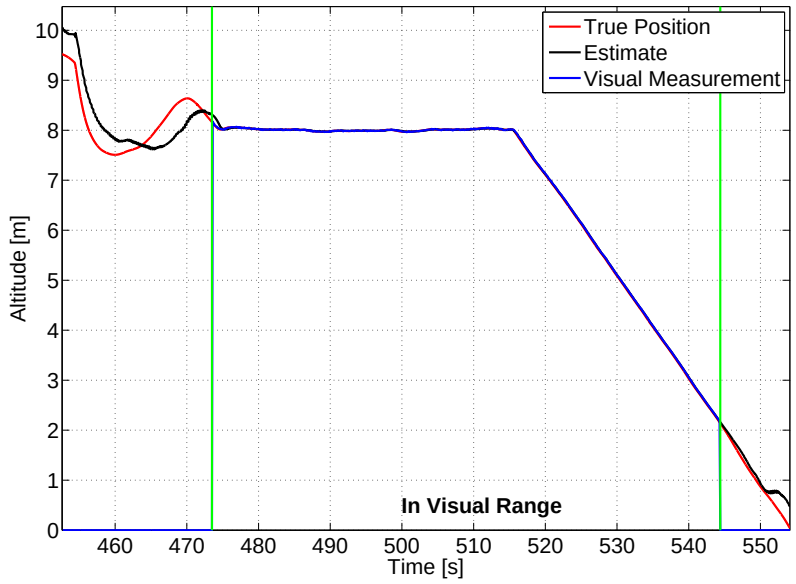


Figure 3.6: Estimator results - Down position

3.8 Summary

In this chapter the simplified estimator was augmented to allow the fusion of vision based position and attitude measurements (Sections 3.2 to 3.4). The method of performing measurements updates in different sensor domains was discussed in Section 3.5. A smooth

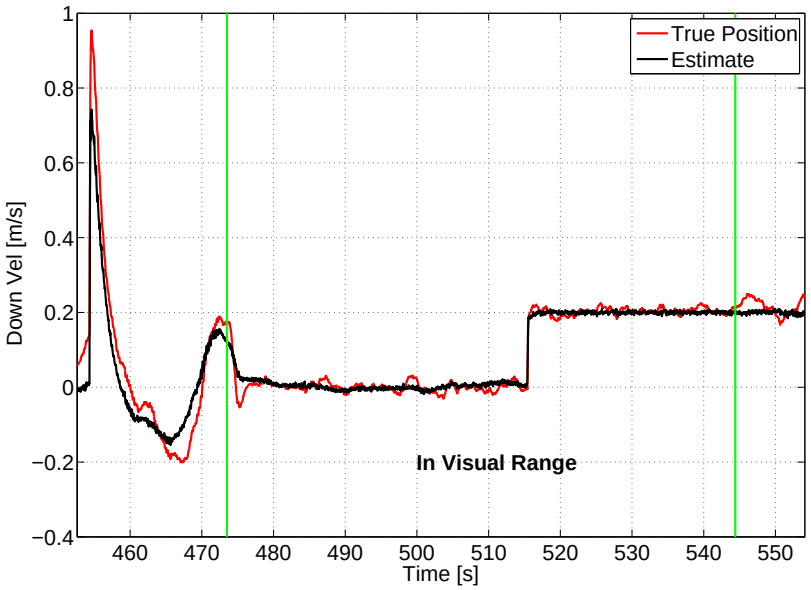


Figure 3.7: Estimator results - Down velocity

transition method was derived to prevent sensor domain transitions from causing steps in the state estimates which could lead to undesired steps in the control commands. The functionality of the estimator was confirmed by the simulation results in Section 3.7.

Chapter 4

High-Level Guidance Algorithms

4.1 Introduction

As mentioned in Chapter 1, the goal of the ATOL research projects at the ESL is to perform autonomous landing on a moving platform that simulates a ship's deck. In this project the landing is limited to a stationary platform. In the previous chapters the necessary algorithms and systems were developed to obtain high-accuracy position and attitude measurements relative to the landing target. In this section high-level guidance algorithms are developed to guide the helicopter around a circuit defined by waypoints and also perform a hover-based landing. The high-accuracy estimates resulting from the visual feedback are used during the landing procedure.

In this project all control loops up to and including the position loop are referred to as inner loops (since they are lower in the controller hierarchy than the guidance algorithms). It is also assumed that the controllers in these loops allow the helicopter to be commanded to any position, velocity and heading angle. See [69] for a detailed explanation. The guidance algorithms are then responsible for generating position, velocity and heading angle references [88].

The waypoint navigation algorithm is outlined in Section 4.2 and the landing algorithm is described in Section 4.3.

4.2 Waypoint Navigation Algorithm

The waypoint navigation algorithm is responsible for generating the position, velocity and heading references that guide the helicopter around a circuit defined by waypoints.

4.2.1 Algorithm

A circuit can be considered as a sequence of flights between two specific waypoints. As a result, the algorithm below is a repetition of steps that guide the helicopter between the

current waypoint and the next waypoint:

1. When the helicopter reaches a waypoint, it hovers for a predetermined amount of time.

2. When the required amount of time has passed, the heading is changed to face the next waypoint before it hovers at that waypoint for another predetermined amount of time.

3. When the required amount of time has passed, the helicopter moves towards the next waypoint at a constant longitudinal velocity while staying in the vertical plane defined by the two waypoints (Figure 4.1). At the same time the altitude is controlled to the destination waypoint's altitude while the heading angle is maintained.

4. When the helicopter is close to the destination waypoint, the constant longitudinal velocity is exchanged for a longitudinal velocity that is proportional to the error between the current position and the destination waypoint (by arming the position feedback loop). This is to ensure that the helicopter does not have significant longitudinal velocity when reaching the waypoint and result in the helicopter overshooting the waypoint. When the waypoint is reached the steps are repeated for the next part of the circuit.

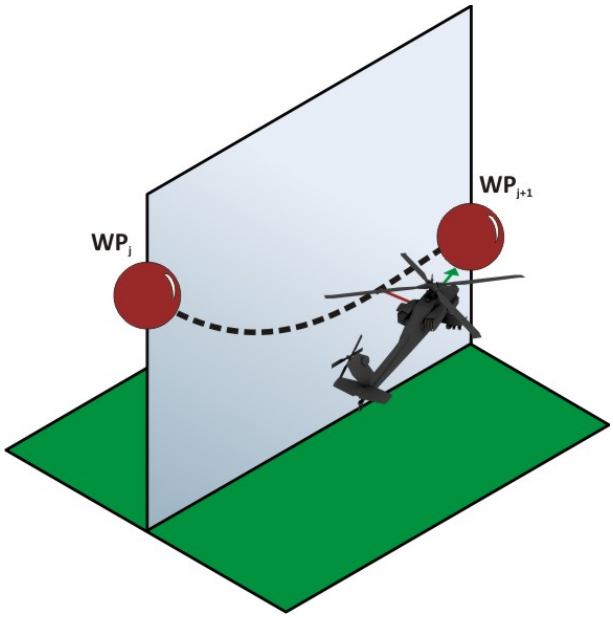


Figure 4.1: The plane defined between two waypoints

These steps are repeated for all the pairs of successive waypoints in a circuit. When the navigation procedure is started the heading is changed to face the waypoint before the helicopter moves towards the first waypoint at constant longitudinal velocity using the same approach as outlined in step 3 above. The navigation controller can be stopped at any time: if the landing controller is activated a landing will be performed, otherwise the helicopter will hover at the position where the navigation was stopped.

4.2.2 Lateral Error Regulation

Without considering the altitude, the problem of regulating the helicopter to the plane can be represented by the 2-D simplification in Figure 4.2.

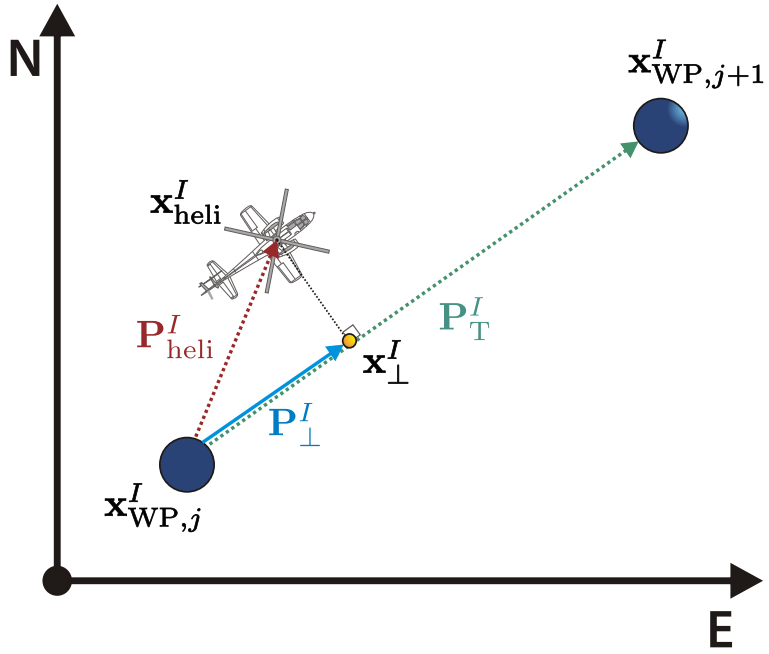


Figure 4.2: A 2-D simplification of guiding the helicopter to a path

For this derivation it is assumed that the current waypoint's coordinates are given by $\mathbf{x}^I_{\text{WP},j}$, the destination waypoint's coordinates by $\mathbf{x}^I_{\text{WP},j+1}$ and the helicopter's current position by $\mathbf{x}^I_{\text{heli}}$. $\mathbf{P}^I_{\text{T}}$ is a vector between the waypoints and $\mathbf{P}^I_{\text{heli}}$ is a vector between the current waypoint and the helicopter.

The helicopter can be guided back onto the line by commanding it to $\mathbf{x}^I_{\perp}$ which is calculated by Equation 4.1:

$$\begin{aligned} \mathbf{x}^I_{\perp} &= \mathbf{x}^I_{\text{WP},j} + \mathbf{x}^I_{\perp} \\ &= \mathbf{x}^I_{\text{WP},j} + \frac{(< \mathbf{P}^I_{\text{heli}}, \mathbf{P}^I_{\text{T}} >) \cdot \mathbf{P}^I_{\text{T}}}{|\mathbf{P}^I_{\text{T}}|^2} \end{aligned} \tag{4.1}$$

where $< \mathbf{P}^I_{\text{heli}}, \mathbf{P}^I_{\text{T}} >$ represents the inner product of $\mathbf{P}^I_{\text{heli}}$ and $\mathbf{P}^I_{\text{T}}$.

Since the helicopter is commanded at a constant longitudinal velocity while the lateral position error is regulated, the coordinates for $\mathbf{x}^I_{\perp}$ are updated at each time step. The rate at which the lateral position is regulated is proportional to the magnitude of the error and is determined by the gain in the position inner loop. In a similar manner, the altitude is controlled by simply updating the altitude controller's reference to the altitude

of the destination which results in the descent/climb rate effectively being determined by the gain in the position inner loop. In general this is not the ideal solution and the altitude should also be regulated to a track between the waypoints. For this thesis, however, the waypoint system is just to provide a framework from which a landing could be demonstrated, so the simplest approach is used.

4.2.3 Implementation

The waypoint navigation algorithm is implemented using a finite state machine to control the different phases mentioned in Section 4.2.1. A diagram of the state machine is shown in Figure 4.3.

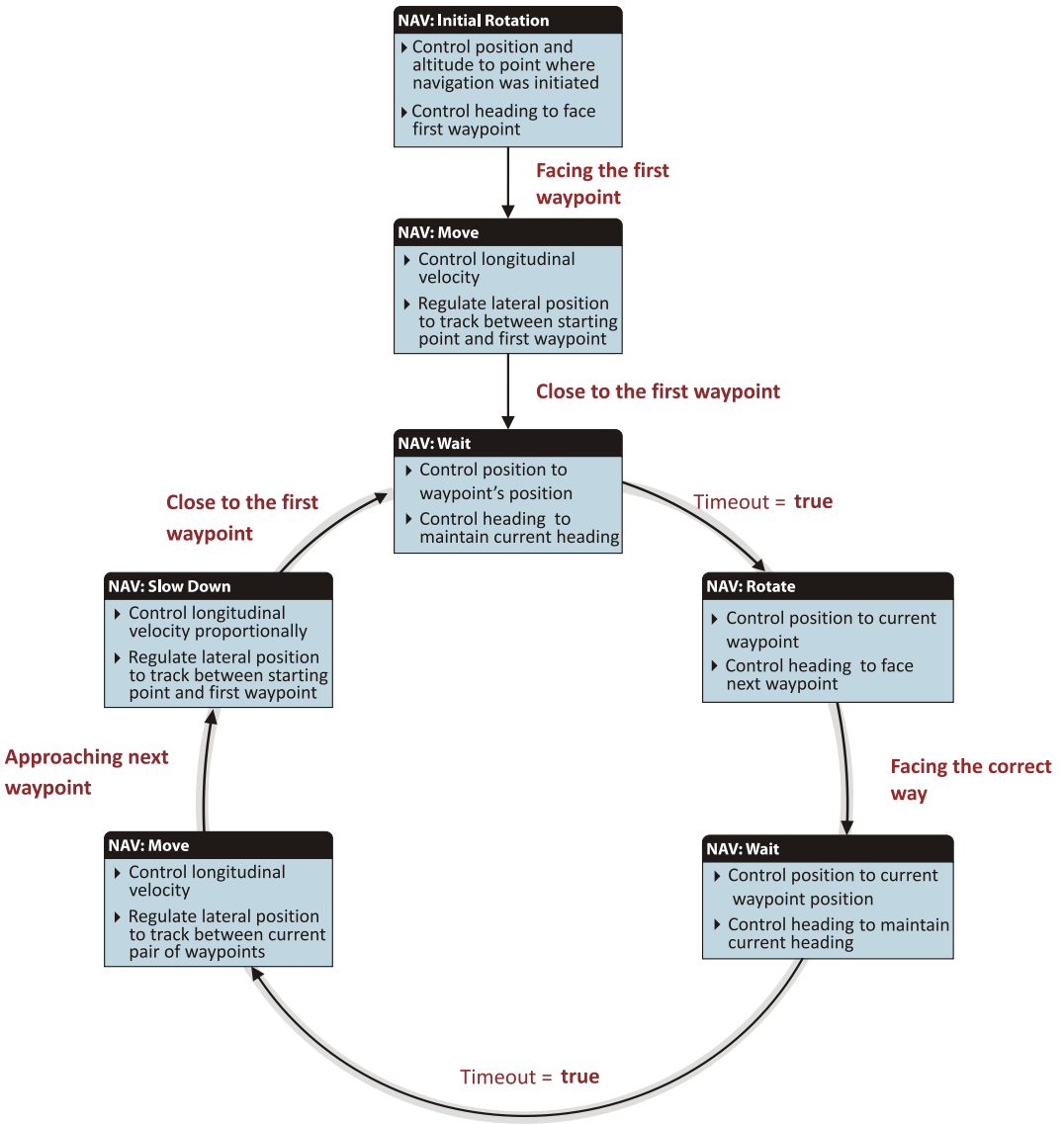


Figure 4.3: The finite state machine governing the waypoint navigation procedure

4.3 Landing Algorithm

The landing algorithm is responsible for calculating the position, velocity and heading angle references necessary to perform a vertical landing on a stationary platform at a known location.

4.3.1 Algorithm

The landing manoeuvre is performed through the execution of the following steps:

1. When the landing command is given, the helicopter's heading reference is adjusted to face the hover point directly above the landing target.

2. Once the heading angle is acceptably close to the reference, a longitudinal velocity reference is applied to move towards the hover point while regulating the lateral error to the track between the point where the landing was initiated and the hover point. This is done using the same approach introduced in Section 4.2.2. The altitude reference is also adjusted to the hover point's altitude.

3. When the helicopter is in close proximity to the hover point, a heading command is given to face in a predetermined direction. During this stage the visual lock is obtained and the estimator will start converging on the visual measurements. During the heading change, the temporary imbalance in side force could cause the helicopter to drift and lose visual lock. This is of no concern since the estimator has guaranteed smooth estimates due to the smooth-transition method in place.

4. Once the heading angle is acceptably close to the reference, the hover is maintained for a predetermined amount of time after which it goes into an alignment phase.

5. During the alignment phase visual lock should have been obtained and the alignment phase effectively waits for the state estimates to converge to within acceptable bounds.

6. When the required position accuracy has been obtained, a constant descent rate is commanded while still regulating the translational position to the imaginary vertical line between the hover point and the landing target using cyclic adjustments. The heading angle is also still controlled to the predetermined reference during the descent.

7. If visual lock is lost during the descent, before the altitude where it is expected to happen, the helicopter is returned to the hover point since accurate measurements cannot be guaranteed. The alignment phase is re-entered and the descent is repeated once sufficient accuracy has been obtained.

8. When touchdown has been achieved the throttle is closed. Touchdowns can be detected by limit switches mounted on the skids or considering sudden changes in the normal accelerometer measurement.

4.3.2 Implementation

The landing algorithm is also implemented using a finite state machine shown in Figure 4.4.

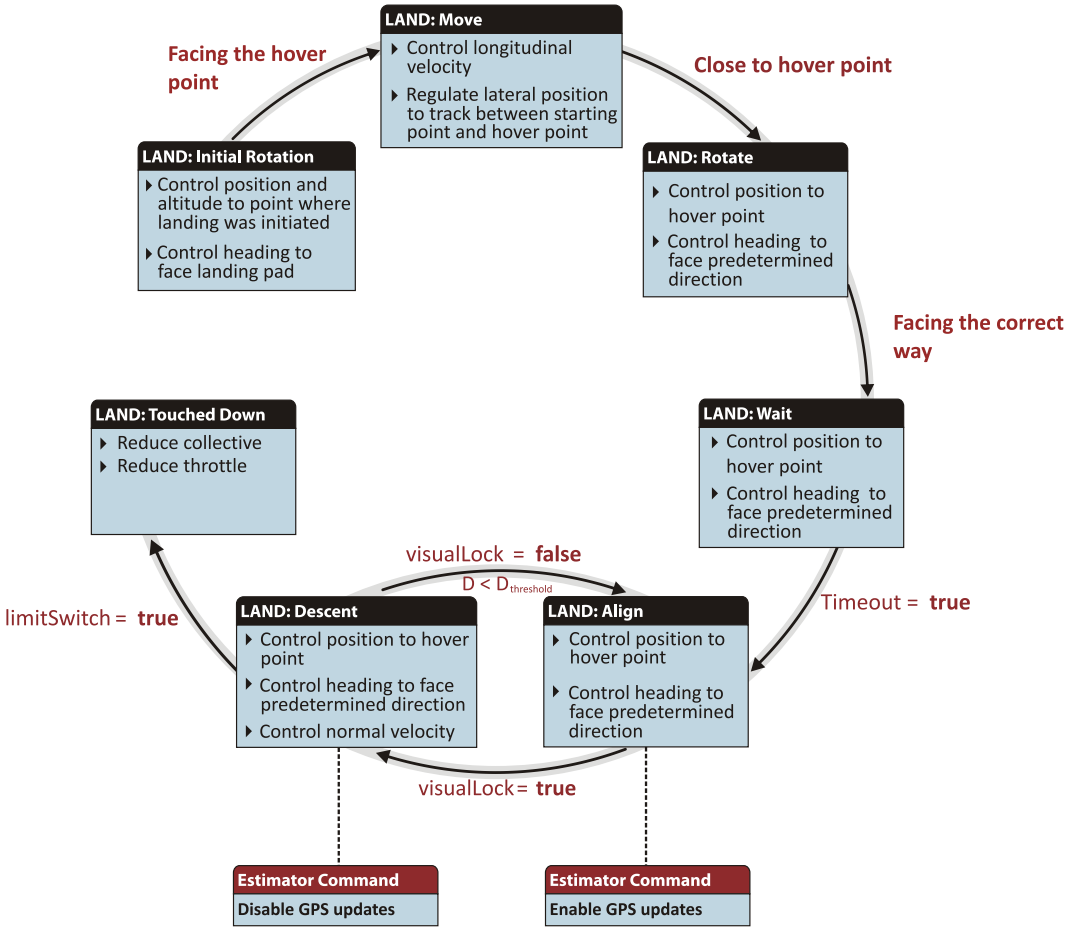


Figure 4.4: The finite state machine governing the landing procedure

4.3.3 Ground Effect

Ground effect is the reduction of induced flow when a helicopter approaches a surface [93]. The resulting effect is an increase in angle of attack which increases the total lift force without increasing the blade angle or the rotor speed. To maintain a constant descent rate a further reduction in collective is required. Since this is an aerodynamic effect, it is assumed to be handled as a disturbance by the inner loop controllers and will not be discussed in detail. However, the ability of the heave loop to reject a disturbance such as this will be demonstrated. This is done by adding a small constant upwards force to the helicopter to simulate the effect of additional lift when it has reached an altitude equal to one blade length [93].

The altitude during the descent phase is shown in Figure 4.5.

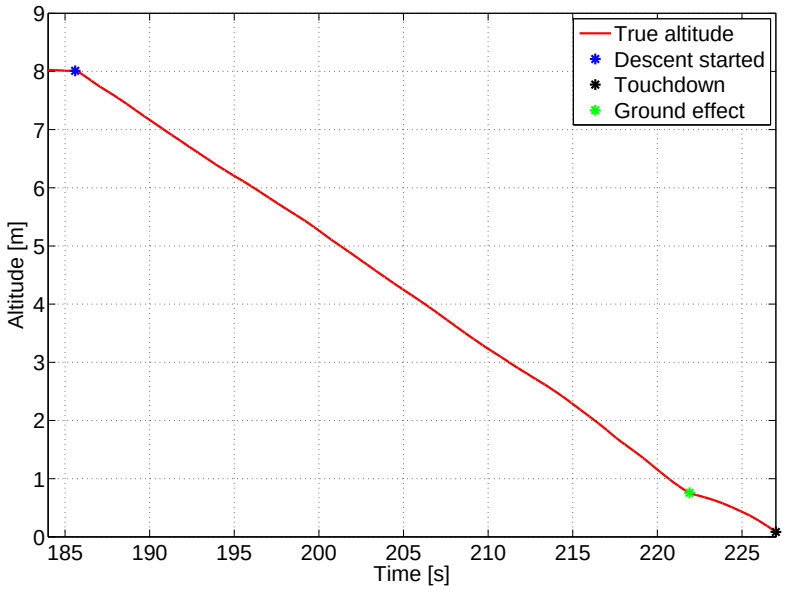


Figure 4.5: Altitude during the descent phase

The reduction in descent rate at the moment when the disturbance force is introduced is shown in Figure 4.6. The collective command to counteract this reduction is shown in Figure 4.7. Since less lift is now required to maintain the same descent rate, the amount of collective is reduced. The heave inner loop is thus able to reject disturbances caused by additional forces due to proximity to the ground and able to maintain the desired descent rate.

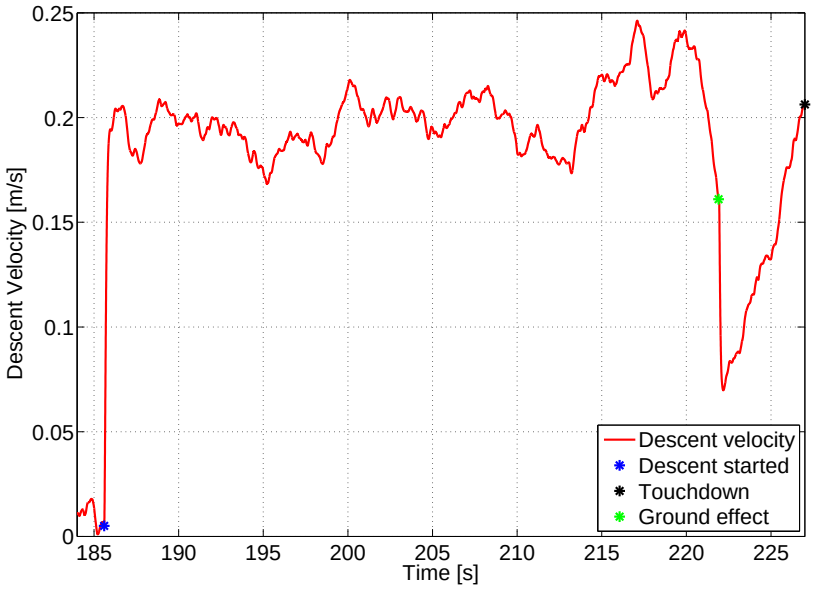


Figure 4.6: Descent velocity the descent phase

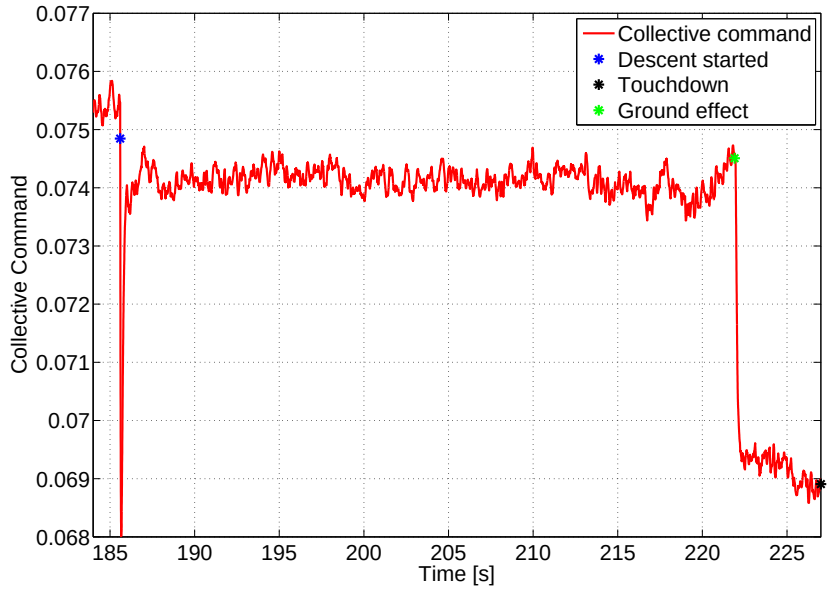


Figure 4.7: Collective command during the descent phase

4.4 Summary

In this chapter high-level guidance algorithms were developed to achieve waypoint navigation (Section 4.2) and autonomous landing (Section 4.2). The inner loop designs are not considered in this project, but the ability of the heave controller to reject an external disturbance such as ground effect was tested, and confirmed the robustness of the inner loops developed in [69]. The results of the navigation and landing algorithms are presented in Chapter 6.

Chapter 5

Vision Hardware Design and Implementation

5.1 Overview

The development of a low power, integrated VPAM (vision-based position and attitude measurement) node forms a crucial part of this research project since image processing is normally associated with powerful hardware and the aim of this project is to achieve it with low-cost, low-power hardware. To achieve the image processing on small, low-power hardware, specialised algorithms were developed for this specific application in Chapter 2. This chapter provides a high level overview of the hardware design as well as a more in-depth look at the algorithms required to perform the image acquisition, processing, matching and pose estimation. To achieve image processing on devices with limited computational ability, a highly integrated approach is required between the hardware and firmware development. A high level overview of the hardware design is shown in Figure 5.1.

A detailed overview of the component design is given in Section 5.2, while the firmware design is discussed in Section 5.3.

5.2 Component Selection

This section outlines the design decisions made in choosing the components. In addition to complying with the specifications, availability and the ability to be hand soldered (to save production cost and simplify debugging) are also considered.

5.2.1 Image Sensor

The image sensor is responsible for capturing an image of the landing target that will be processed by the algorithms developed in Chapter 2 to determine the pose and position

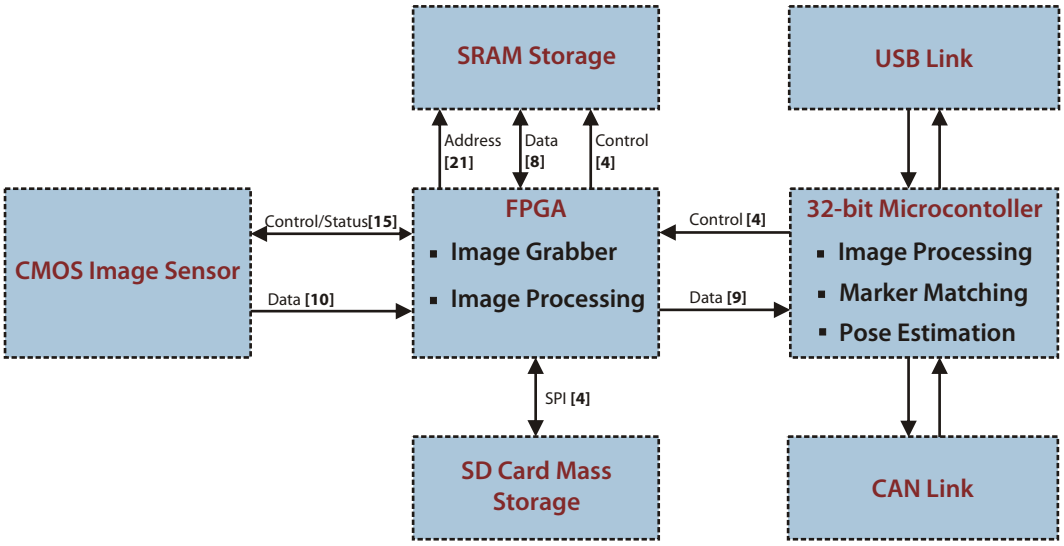


Figure 5.1: Overview of the VPAM hardware

of the vehicle relative to a known landing target. Given the application of the system, a number of requirements for the image sensor can be determined.

Pixel Dimensions

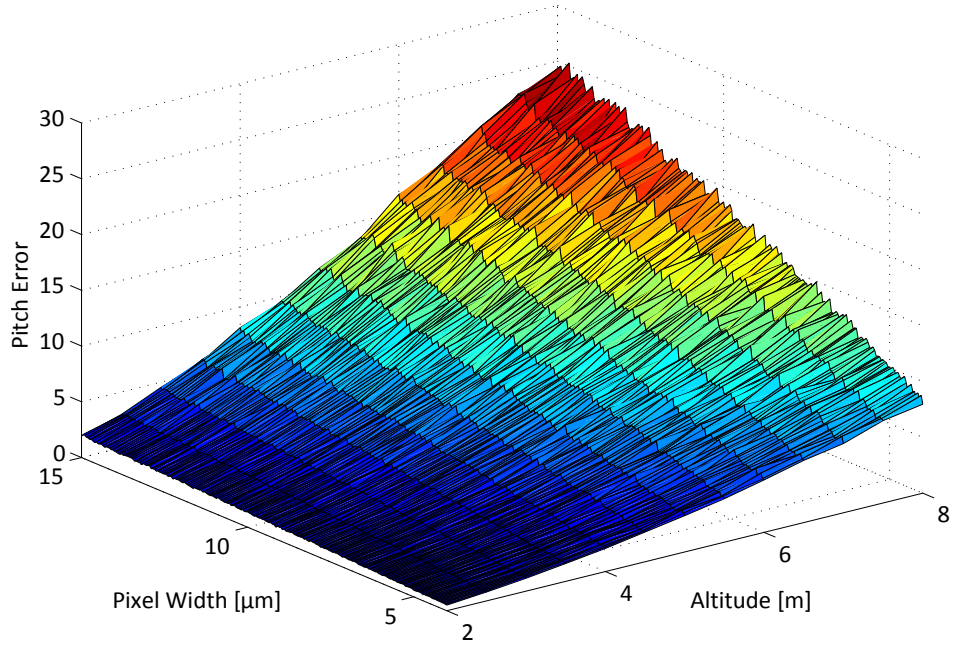
The discrete nature of the image sensor can have a very direct impact on the accuracy of the proposed pose estimation algorithm. The angular accuracy at varying altitudes with varying pixel dimensions is shown in Figure 5.2a while the position accuracy is shown in Figure 5.2b. In these plots the accuracy is determined by considering the error between the theoretical estimate (using an image sensor with no noise and infinitely small pixels) and a simulated image from the image sensor. The simulated image is created by quantising the projection on the image plane and simulates a worst case scenario where the projection covers only a single pixel. For practical markers, the projection would cover a much larger area and the effect of discritisation is diluted, but it still provides a lower bound on accuracy especially at high altitudes when the marker projection is small.

From Figure 5.2 it can be seen that:

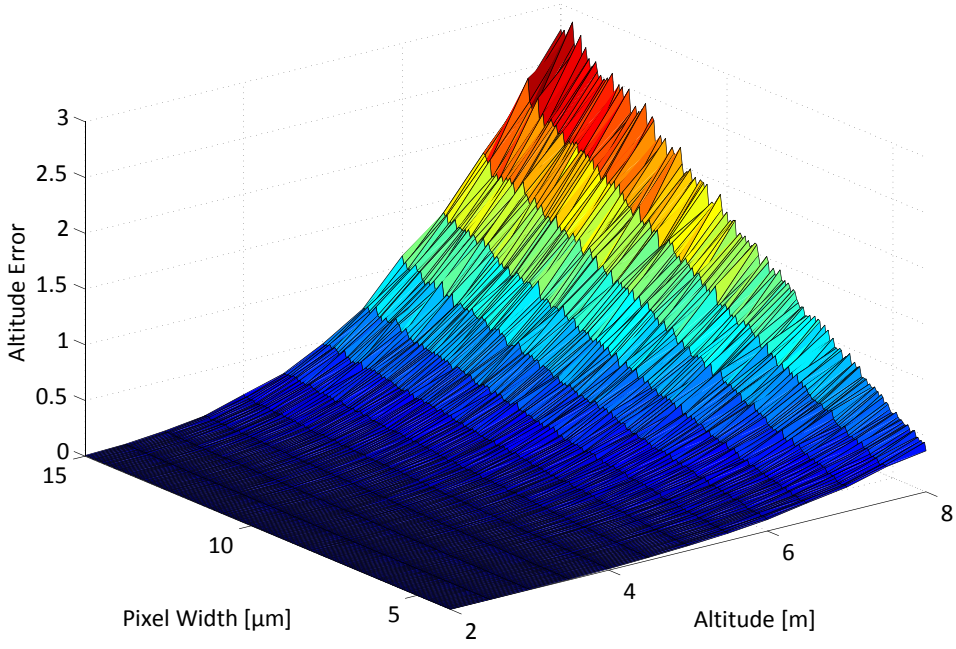
1. Accuracy decreases with increasing altitude as the effect of quantisation has a more significant effect when the projection of the landing target has little separation between markers.

2. Accuracy decreases with increasing pixel dimensions. This is simply because a larger part of the real image plane is discretised to the same pixel, leading to a larger error which directly influences the unit vector calculated by the pose estimation algorithm.

From an availability perspective three image sensors by Cypress Semiconductor were considered:



(a) Pitch Angle accuracy



(b) Altitude accuracy

Figure 5.2: The effect of discritization on pose estimation accuracy

From the surface plots shown in Figure 5.2a and Figure 5.2b, it is clear that smaller pixels are desired to improve accuracy (especially at higher altitudes). To limit the angular error at the initial hover point altitude, the CYII5SM1300 is the best choice of the available sensors. For a 6.7 μm pixel width, the upper bound of the altitude error is 0.95 m and the angular (roll and pitch) error is 10.32°. These errors decrease with decreasing altitude as the centroids spread across the sensor and the effect of discritisation become less pronounced. Translational position and yaw angle are less sensitive to quantisation and are not shown.

Table 5.1: Candidate image sensors

Model	Number of Pixels	Pixel Dimension	Shutter Type
CYIL1SM0300	640×480	9.9 μm	Global Only
CYIS1SM1000	1024×1024	15 μm	Rolling Only
CYII5SM1300	1280×1024	6.7 μm	Global and Rolling

Shutter Type

A rolling shutter is a method of image acquisition where the frame is exposed row by row and is, as a result, not a snapshot of a single point in time. This method is typically used in CMOS sensors and results in image skew if successive rows are not read out in quick succession. A global shutter (or synchronous) shutter makes use of additional logic on each pixel to expose the entire image sensor at the same point in time which results in no image skew. Since the design involves pipelined processing of each row, there might be a significant delay between the readout of successive rows which will lead to distortion. This places a severe limitation on the processing time for each row, thus a global shutter is preferred. This specification is also met by the CYII5SM1300 sensor.

Spectral Response

During the derivation of the algorithms in Chapter 2 it was assumed that the image will be high contrast and generally only contain the vertices of the landing target. To obtain such an image, lights (with significant power in the infrared (IR) band) are used as markers to indicate the landing target. A narrow band interference filter is mounted in front of the lens to filter out visible light and just allow the transmission of IR light. Since IR light falls outside the visible light spectrum, most image sensors filter out IR light to prevent a reduction in dynamic range, but for this application high IR sensitivity is desired. Unfortunately sunlight has significant power in the non-visible (UV and IR) bands which could cause disturbances or saturation of the image sensor. The likelihood of this can be reduced by choosing the wavelength of the marker lights to be equal to a minimum in the solar radiation spectrum (Figure 5.3).

Two practical choices are 950 nm and 1100 nm since high powered artificial lights at higher wavelengths are not commonly available and will not fit within the low-cost framework of this project. In addition to choosing the markers to radiate power in a certain part of the spectrum to reduce the likelihood of disturbances, the chosen image sensor must have significant sensitivity at the chosen wavelength. A comparison of the spectral responses of the available sensors is shown in Figure 5.4.

The CYII5FM1300 sensor is a revision of the CYII5SM1300 (that satisfies the other specifications) that has double the sensitivity of the other sensors in the 950nm band. This, combined with the availability of 950nm sources, make it the ideal sensor for the camera node.

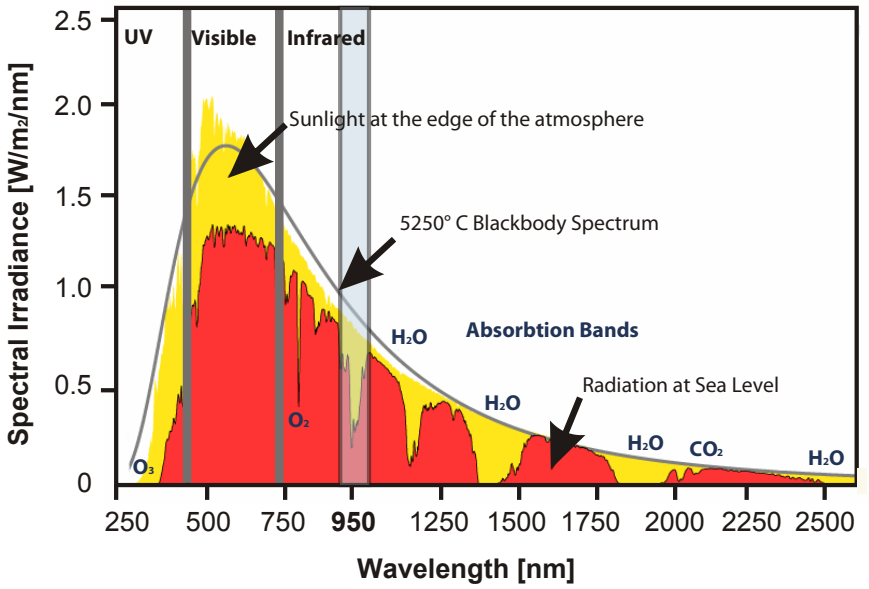


Figure 5.3: Spectral irradiance of sunlight

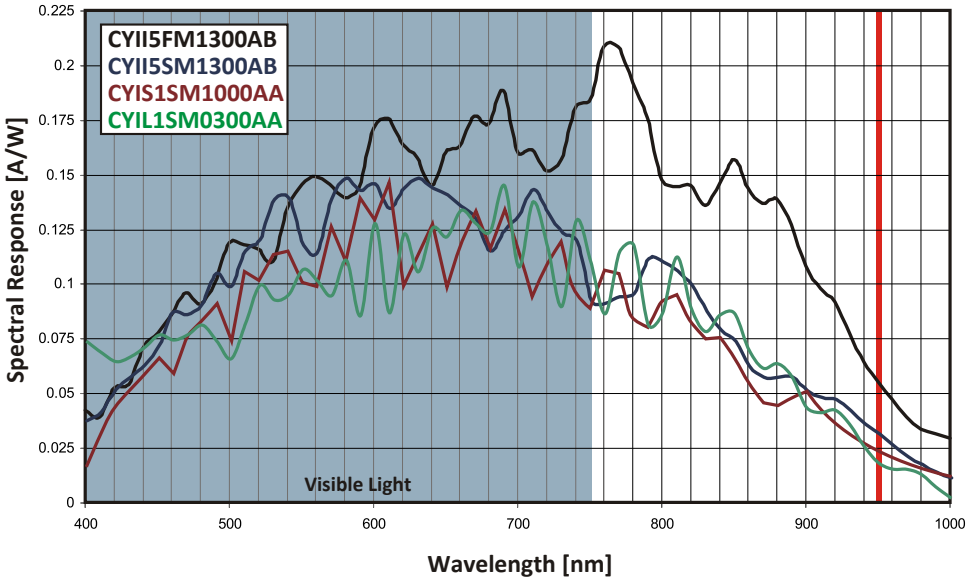


Figure 5.4: Spectral comparison of image sensors

It is the same sensor used in [92], chosen based on the higher IR response at 950nm as well as the horizontal resolution.

5.2.2 Processing and Interfacing

Field programmable gate arrays (FPGAs) are often used as glue logic since their architecture makes them ideal for providing an interface between different components in a system. Due to the high density of modern logic devices they are able to implement high complexity functions and even emulate microcontrollers. They are also reconfigurable which provides a high level of flexibility in a prototype design. As a result, an FPGA forms the core of the camera node.

The matching and pose estimation algorithms developed in Chapter 2 utilise many mathematical operations; something that is not a native/direct capability of an FPGA. A microprocessor is suited much better for such operations. As mentioned in the previous paragraph, it is possible to implement a (softcore) microcontroller within the FPGA, but there are two significant downsides to such an approach (for a research project):

1. Cost of the license to implement softcore processors versus the cost of a dedicated microprocessor. A typical 32-bit microcontroller costs around R40 in small quantities. The license to implement a softcore with a similar number of peripherals (if they are not custom developed, but rather bought as intellectual property (IP)) is significantly more. Some manufactures add a license fee to each FPGA sold which dilutes the cost (Actel), while others require a yearly license to implement the softcore and associated IP peripherals (Altera).

2. A softcore processor requires a large number of logical elements. On the small capacity FPGAs that can be hand soldered, this will have a noticeable impact on the number of other logic functions that can be implemented. If a dedicated microprocessor is added to the design, more freedom is available with regards to the FPGA development, but with the added option of a softcore processor if required by the design.

As a result of these two factors, the decision was made to use an FPGA for interfacing and pipelined processing (thresholding and compressing) and a microcontroller for the more intensive image processing (matching and pose estimation).

FPGA

Two manufacturers dominate the FPGA market: Altera and Xilinx [1]. Since Altera devices have been used most recently in the ESL [9, 92]) and the availability of development tools and practical experience gained during undergraduate courses made it the obvious choice. Since the FPGA will be used to interface with the CMOS image sensor, the maximum required IO frequency is 40 MHz [14]. The Altera Cyclone II series (-7 speedgrade) has a maximum internal frequency of 500 MHz and IO toggle rate of 300 MHz [3] which is adequate for interfacing with the image sensor and implementing the computational logic.

There are seven devices in the Cyclone II family with the same performance characteristics, but varying amounts of embedded memory block, logic elements and I/O pins. Only two of the devices are available in a hand solderable TQFP package Table 5.2:

Both devices have the same physical dimensions, and the additional logic elements and embedded memory of the EP2C8 will allow more freedom for future upgrades. The resource utilisation of the final design is shown in Table 5.3.

Table 5.2: Resource comparison of available FPGAs

Device	Logic Elements	Memory (Kbits)	IO Pins (TQFP)
EP2C5	4608	117	142
EP2C8	8256	162	138

Table 5.3: Resource utilisation of the final design

Total Logic Elements	1040/8256 (13%)
Total Pins	63/138 (46%)
Total Memory Bits	30720/165888 (19%)

Microcontroller

A Microchip device is used as all the ESL avionics utilise microcontrollers manufactured by Microchip and all the development tools are readily available. The PIC32 had recently been launched when the hardware design was started and has a very powerful core running at 80 MHz. The PIC32 contains most of the peripherals available in the dsPIC30 and dsPIC33 families, with the only exception being a controller area network (CAN) module[1]. Since it is based on 32-bit architecture, porting algorithms from the PC (after thorough testing) is very convenient.

The PIC32MX360F512L was chosen for two reasons: It was already available in an 80 MHz revision (as opposed to the release configuration of 72 MHz) and had a generous amount of program memory (512 KB) and RAM (32 KB). A linked-list structure of the compressed image data is constructed during the readout and more RAM allows a larger heap which allows more image data to be handled resulting in a more robust system.

5.2.3 Communication

Two communication interfaces are implemented in the camera design: USB and CAN. The USB interface is used for debugging, development and downloading image data while the CAN interface is used for interfacing with the ESL avionics system during flight. During normal operation the USB interface is not required.

USB

When a large amount of data must be transferred between the camera node and PC (for debugging and analysis) a USB interface is preferred over a conventional serial port due to the higher speeds and reliability of the differential approach used. USB is also more prevalent since integrated serial ports on new PCs are deprecated and although USB to

[1] A PIC32 with CAN was released in 2009, after the hardware was designed. Despite this, there are advantages to the approach taken. See Section 5.2.3

Serial converters provide basic functionality, they tend to be very unreliable at high transfer rates. The FTDI FT232R device provides USB connectivity with a serial interface. This simplifies both the hardware (since the microcontroller has an integrated serial interface) as well as the software running on the PC since the USB port is enumerated as a (virtual) serial port. Using the royalty free virtual COM port driver, transfer rates of up to 1 Mbps can be achieved. Downloading the full resolution image will take ten seconds at this rate, which is a significant improvement over the rate that is practically achievable with a conventional serial port.

CAN

CAN is a bus standard that allows inter-device communication with a host controller. Data is embedded in messages with a unique identifier for each message and arbitration is performed by the controllers without intervention from the embedded firmware. This bus is already used in the ESL avionics system and provides a very simple interface to receive data from the avionics and deliver the results from the pose estimation to the estimator. Since the chosen microcontroller does not have a built-in CAN controller, the MCP2515 stand-alone CAN controller is used. This device has an SPI interface with transfer rates up to 10 MHz and supports the CAN V2.0B specification at 1 Mbps. This is the same specification used in the existing avionics allowing for up to eight data bytes per message in an extended frame. Similar to the CAN controller in lower end PIC devices, it has two receive buffers, six filters and two masks. This allows the device to only process certain messages (programmed via the SPI interface) which alleviates the load on the firmware. A generic output is provided that can be programmed to be asserted under specific conditions (new message available, message error etc.) By connecting this output to an external interrupt on the microcontroller, messages can be received without having to poll the SPI device.

The advantage of using the stand-alone controller is having separate clock domains for processing and communication systems. In the existing ESL avionics the crystal for the microcontroller (9.6 MHz) was chosen to achieve a CAN baud rate of 800 kbps and 19.6 million instructions per second (MIPS) on the processor core. Since the dsPIC is rated for 30 MIPS, the processor is severely underutilised. Changing the crystal or baud rate will affect a large amount of code that has been developed and tested over the last five years. By having separate clock domains, this problem is alleviated and the CAN is still operated at 800 kbps (using an external 16 MHz crystal on the MCP2515) while the PIC32 core is operated at 80 MIPS (using an external 7.3728 MHz crystal on the PIC32).

5.2.4 Storage

Having logged data is indispensible for analysis and debugging. SD cards are available in large capacities (up to 32 GB), at very low prices and have a built in SPI controller which can be easily interfaced to the microcontroller. In addition, with a proper file system implementation on the microcontroller, the log files can be directly read on any computer. In addition to the SPI interface, SD cards also provide a 4-bit parallel-data

interface capable of higher transfer rates. Although this is not implemented, the control and data lines between the microcontroller and the SD card holder are routed through the FPGA. This will allow future expansion where data could be written directly at high speed from the FPGA to the SD card.

Two megabytes of volatile storage in the form of SRAM provides temporary storage for a full resolution image during development and debugging. The chosen device is the CY7C1069DV33 manufactured by Cypress. It has a maximum address- to-data-valid time (t_{AA}) and write-cycle time (t_{WC}) of only 10 ns [13] which is faster than the image sensor's download rate (25 ns per pixel) and will not provide a bottleneck in the design. Data transfer occurs through a parallel interface (8 bits for data and 21 bits for the address) and a set of control lines (chip enable, write enable and output enable). The appropriate commands and timing are handled by the FPGA and discussed in the next section.

To allow the memory to be used for temporary storage during processing on the microcontroller, control lines, address lines and data lines were also routed from the microcontroller to the FPGA. Since the memory on the microcontroller is adequate for the current implementation, it remains unutilised.

5.3 Algorithmic Flow

This section discusses the algorithmic flow of the visual position and attitude sensor, starting at the image capture stage and ending at the return of the position and relative attitude results as shown in Figure 5.5. Before an image is captured, the CMOS image sensor and associated hardware is configured to predetermined settings (Appendix A). While the image is downloaded from the image sensor, it is compressed and processed by the region growing algorithm to determine the coordinates of the high-intensity regions. Due to the pipelined nature the region growing will be complete just after the image download has finished. The matching algorithm (Section 2.5) is then performed to select the regions that most likely represent the projection of the target markers. Finally, the relative attitude and position of the camera to the target is determined by the pose estimation algorithm developed in Section 2.6. A more detailed flow diagram with responsibility assignment (FPGA/Microcontroller) is shown in Figure 5.6 and discussed thereafter.

The following events take place when the microcontroller determines (from a timer event) that an attitude and position measurement should be captured:

1. A series of configuration commands are sent from the microcontroller to the FPGA to configure the image sensor to the desired settings. Details of the configuration commands are discussed in Appendix A.

2. The commands are parsed by the command handler and the configuration is performed by the CMOS configuration handler discussed in Section 5.3.2.

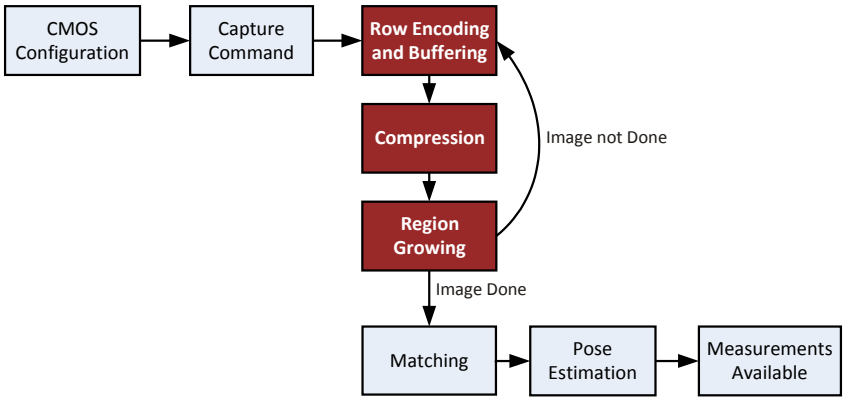


Figure 5.5: Simple algorithmic flow for position and attitude measurement

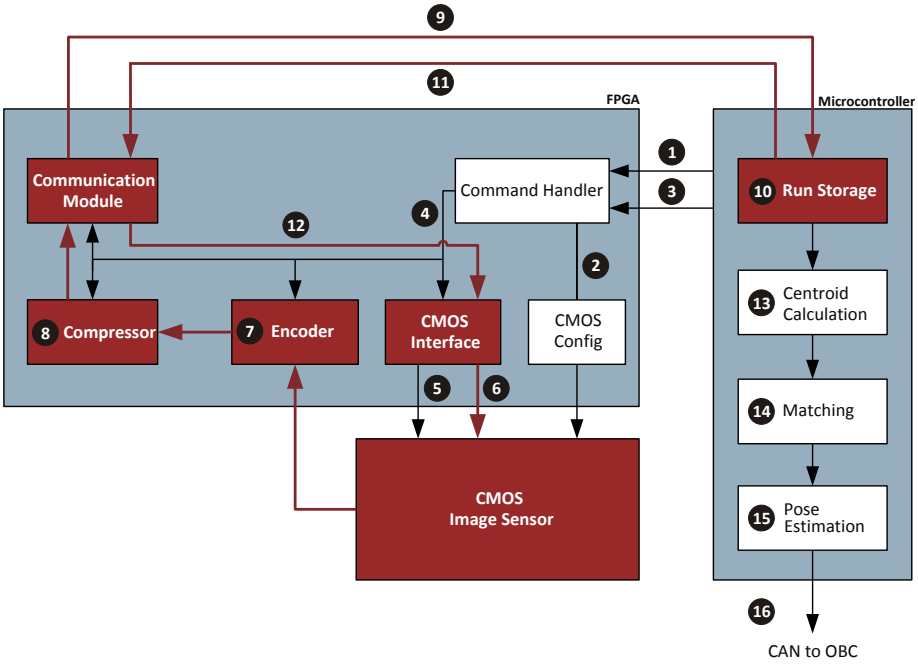


Figure 5.6: Algorithmic flow for position and attitude measurement

3. A command is sent from the microcontroller to the FPGA that requests a new photo to be taken. Details of this communication protocol are discussed in Section 5.3.1.

4. The command is parsed by the command handler and the CMOS interface is notified to start the image acquisition process.

5. Once the image has been captured, image readout of the first row will immediately be initiated by the CMOS interface. It involves a specific sequence of control signals and delays. This module is discussed in Section 5.3.3.

6. When valid pixel data is available, it is encoded and written into a FIFO buffer.

7. As soon as data has been loaded into the input FIFO buffer, the compressor module starts reading the buffer and performing the compression. The compressed version of the row is written into another FIFO buffer.

8. As soon as data has been loaded into the output FIFO buffer, the communications module starts transferring it to the microcontroller. Details of this communication protocol are discussed in Section 5.3.6.

9. The compressed rows are processed as they are received by a region growing algorithm. The result at the end of the image is a list containing information about all the high intensity regions that could be the projection of target markers. Details about the region growing algorithm are discussed in Section 5.3.7.

10. Once the compressed version of the row has been transferred to the microcontroller, the communications module is notified that the microcontroller is not processing and the next row is requested from the image sensor by the CMOS interface and steps 6 to 10 are repeated.

11. Just after the final row has been read from the image sensor, the compressed version of the complete image will be stored in a linked-list structure within the microcontroller's memory and the centroid position and pixel count of high intensity regions (possible landing target markers) can be calculated.

12. The matching algorithm (Section 2.5) is applied to the list of centroids and the best state estimate received over the CAN bus from the OBC to determine the centroids that most likely correspond to the target markers.

13. The pose estimation algorithm (Section 2.6) is applied to the list of likely target marker projections to determine the relative attitude and position.

14. It is packaged in a CAN message and placed on the CAN bus when the next synchronisation pulse is received.

5.3.1 Command Handler

The command handler is implemented on the FPGA as a finite state machine and is responsible for parsing the command and associated data received from the microcontroller. Typical commands change the configuration of the image sensor, set the parameters for the compression algorithm and initiate the image capture. The bus configuration shown in Figure 5.7 is used to transfer the command and associated data

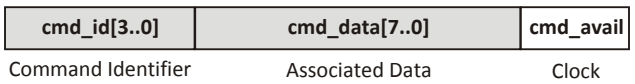


Figure 5.7: Bus used for commands transfer between microprocessor and FPGA

When a command is sent from the microcontroller to the FPGA, the identifier and data are placed on the bus and the command available signal is asserted for two command handler clock periods (200 ns). The 4-bit command identifier and 8-bit data byte is latched by the command handler and then parsed by the state machine in Figure 5.8 (for clarity, the handling states for all the commands are not shown).

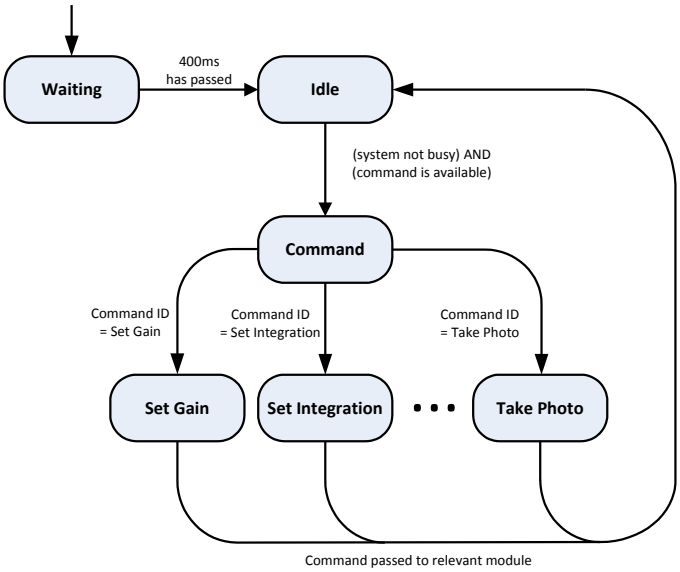


Figure 5.8: Simplified state diagram for the command handler

Table 5.4: Valid command list

Command ID	Associated Data	Command Description
1 (0001)	None	Capture Image
2 (0010)	Gain Index[1]	Set image sensor gain
3 (0011)	Integration Index[2]	Set integration Time
4 (0100)	Value between 0-255	Set threshold
5 (0101)	None	Set image sensor configuration
6 (0111)	None	Reset image

[1] Index to a lookup table with the gain values stored on the FPGA. The table is shown in Appendix A.

[2] Index to a lookup table with the timer values for integration period. The table is shown in Appendix A.

The first state introduces a 400 ms waiting period to reject the transients on the microcontroller output during startup. Commands accepted by the command handler are shown in Table 5.4.

The remaining command identifiers were used during development to configure the SRAM, full resolution image capture and downloading functions. Since these are not needed during normal operation of the camera node, they were removed from the final design.

5.3.2 Sensor Configuration

The sensor configuration module is implemented on the FPGA and is responsible for the configuration of the image sensor through a serial interface. These configuration settings include the gain, integration time, image dimensions and settings controlling the timing and calibration of the ADC and DAC modules. The configuration word (containing

both the address of the register that must be configured and the associated setting) is transferred from the command handler to the configuration module on a 16-bit bus and buffered when the command line is asserted. A finite state machine is used to load the configuration word (in a parallel fashion) into a shift register and shifting it out one bit at a time, while also controlling the serial enable and clock lines. A complete list of the configuration words are provided in Appendix A.

5.3.3 CMOS Interface

The CMOS interface is responsible for performing the capture and managing the image readout. A finite state machine (FSM) is used to generate the relevant control signals in response to internal timing and feedback from the image sensor.

The capture and readout process is a straight forward process detailed in the datasheet and summarised in Figure 5.9. The only exception is indication of valid digital pixel data since the PIXEL_VALID signal, despite its name, does not indicate when the pixel data is valid. During experimentation it was found that five clock cycles after PIXEL_VALID goes high the analog pixel stream at the image sensor's ADC input is valid and after three more cycles the digital representation is available on the DATA_OUT bus. Due to the 8 cycle delay, PIXEL_VALID is not an intuitive signal to indicate when the digital representation is valid. As a result, the CMOS interface generates an internal signal to indicate the validity of data on the internal bus to the following stage in the pipelined process.

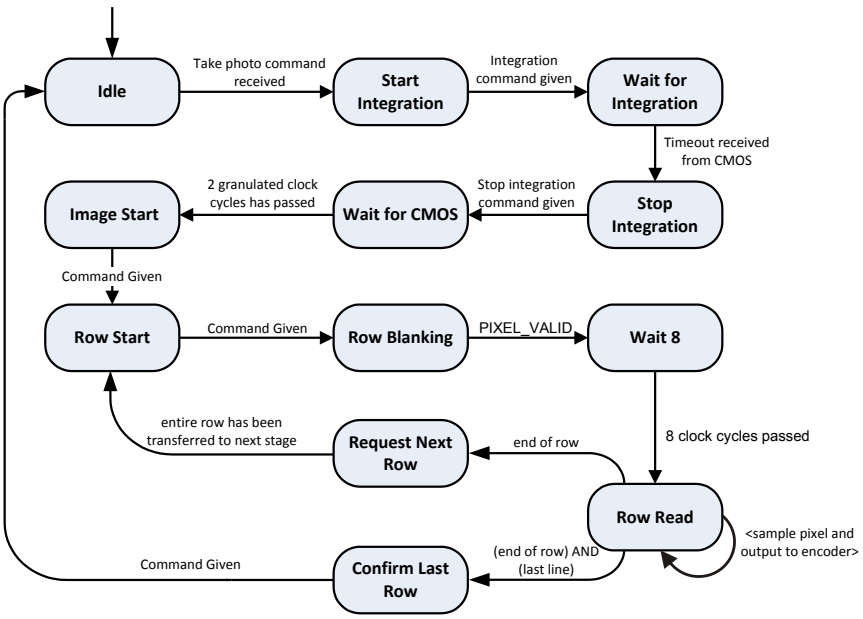


Figure 5.9: Simplified state diagram for the CMOS interface

5.3.4 Encoding

The output from the CMOS interface is a constant stream of 8-bit intensity values for an entire row of pixels. The role of the encoder module is to extract each pixel from the stream and store it in a FIFO buffer for further processing by the compressor module. The extraction and FIFO transfer is managed by a finite state machine which runs at a five times higher clock rate than the CMOS interface. A faster clock (than the CMOS interface) is required to allow the write cycle for the FIFO buffer to complete. A three times faster clock frequency would have been sufficient, but since the compressor module and communication module utilise the same clock, an even faster clock (five times) is used. This allows each row to be processed faster and gives more room for future expansion and modifications. The behaviour of the encoder is shown in Figure 5.10.

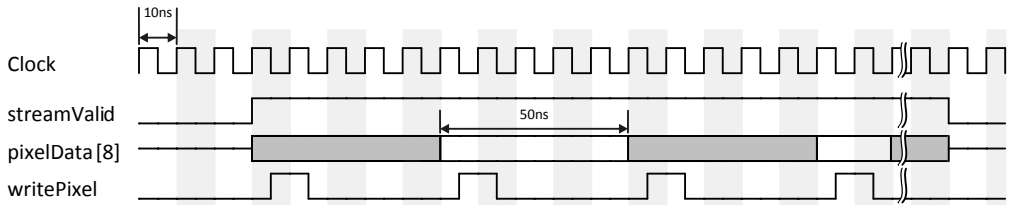


Figure 5.10: Timing diagram for encoder behaviour

5.3.5 Compression

The compressor module is responsible for reducing the amount of information that must be transferred to the microcontroller since the microcontroller has limited memory. Two common compression algorithms were investigated and compared, namely JPEG and PNG. The JPEG compression algorithm is based on the human perception of the image and it transforms sections of the image to the frequency domain and reduces the accuracy of representation of the higher frequency components that will not be very perceptible to the viewer [16]. PNG is an image compression algorithm that compresses an image through two stages: pre-compression/filtering and compression using duplicate string elimination where common sequences are not duplicated, but rather back-referenced and Huffman Coding where commonly-used symbols are replaced with shorter representations and less commonly used symbols replaced with longer representations [16].

The common problem with both these algorithms is their relatively high level of computational complexity. This seemed unnecessary for such simple, high contrast images and the second step of the PNG algorithm prompted research into Huffman coding where common subsequences are replaced with shorter symbols. The simplest version of Huffman coding is called run length encoding (RLE) where sequences of pixels that have similar intensities (referred to as runs) are counted and stored as a position and count instead [77]. This method also allows for an efficient implementation in a pipelined architecture. Due to the infrared markers and bandpass filter, the images will be primarily black with high intensity regions indicating the markers which results in a very high compression ratio and images that can easily be manipulated by a small microprocessor.

Algorithm

The algorithm is executed in a pipelined fashion and involves every pixel being investigated as it is read from the image sensor. If the pixel is above a certain intensity threshold a counter is started and incremented for each successive pixel that also exceeds the threshold. The count is terminated when a pixel that does not satisfy the threshold is encountered. Each time such a run (of high intensity pixels) has ended a package containing the row number, number of high intensity pixels and the starting column of the run is written into the transmission FIFO buffer. This process is repeated for every pixel in the row. At the end of a row a sentinel package is written into the buffer that notifies the microcontroller that a row has ended.

If a row contained no runs, the sentinel is not written and the next row is automatically requested from the image sensor. This deviates from the case where the communications module (Section 5.3.6) requests the next row once the microcontroller has acknowledged receipt of an entire row's run information. It is implemented in this way to allow the FPGA to quickly skip large portions of the image that contains no useful information without intervention from the microcontroller.

At the end of the image, another sentinel package is written into the buffer that notifies the microcontroller that the image has ended. The compression procedure is implemented on the FPGA as an FSM (simplied diagram in Figure 5.11).

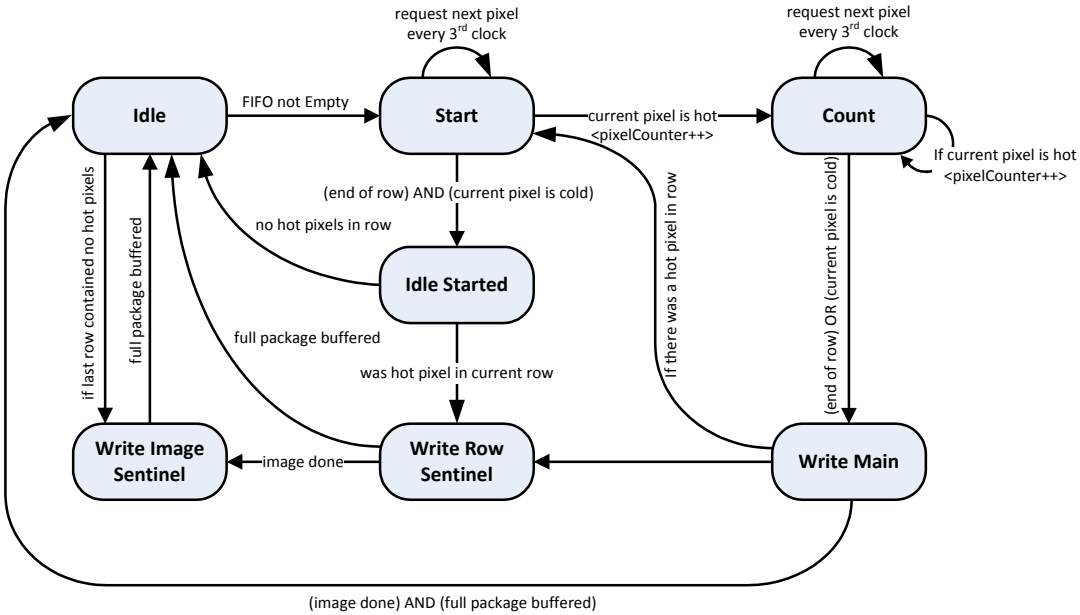


Figure 5.11: Simplified state diagram of compressor algorithm (States implementing the buffer read and write processes are not shown.)

The data for each run is packaged (Figure 5.12) when it is written to the buffer. The package is bookended by two synchronization characters to verify the validity of the package upon receipt on the microcontroller.

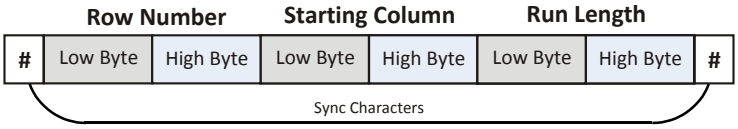


Figure 5.12: Structure of the package containing run data

5.3.6 Interprocessor Transfer

The compressor module transfers run information to the transmission FIFO buffer when a run has ended. This buffer is read by the communication module that transfers the run information to the microcontroller where a compressed version of the image is stored.

An 8 bit data bus and two control lines are used for transmission from the FPGA to the microcontroller. The one control line indicates the status of the microcontroller to the FPGA while the other indicates to the microcontroller that data is available on the bus.

Transmission from FPGA

As a means of flow control, the FPGA will not transfer data if the microcontroller is still busy processing data that has already been received. If the microcontroller is not busy, the communication handler will place data on the bus and pulse the clock line that causes an interrupt to occur on the microcontroller and read the data on the bus. The change notification module on the microcontroller is used to monitor the clock line and detect the rising edge. The finite state machine that controls the behaviour of the communication module is shown in Figure 5.13.

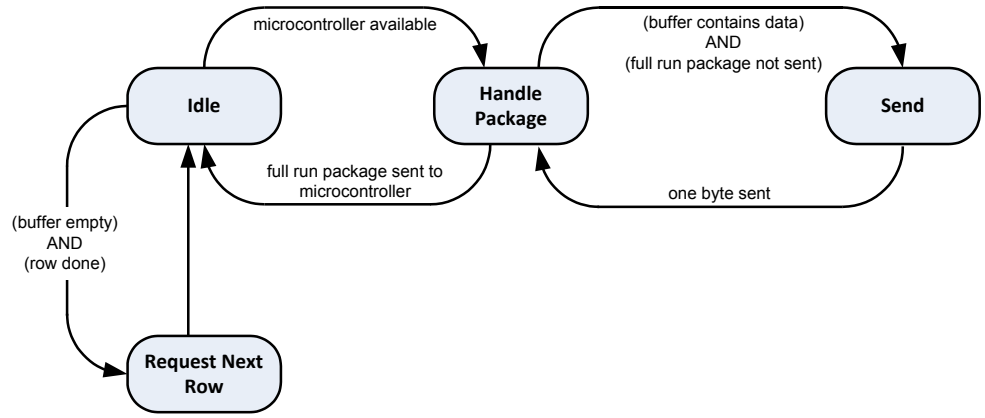


Figure 5.13: Simplified state diagram of communication module on the FPGA

Reception on Microcontroller

Another finite state machine is used on the microcontroller to receive and process the run data received from the FPGA. This process is shown in Figure 5.14. When the interrupt occurs the byte is read from the bus and placed in a buffer. When a complete package has been received, the busy signal is asserted. If it is not a sentinel package, memory is allocated and the run information is added to the linked-list that represents the current

row. The busy signal is reset. When the sentinel package has been received to indicate the end of the row, the current row and previous row is processed by the region growing algorithm to determine regions of high intensity (Section 5.3.7). After the region growing the busy signal is reset which indicates to the FPGA that the next row can be requested from the image sensor. When the end-of-image sentinel package has been received, the areas identified by the region growing algorithm are processed to determine the centroids (Section 5.3.7). Due to pipelined nature of the algorithm, the reception and region growing is performed by the same state machine (Figure 5.14).

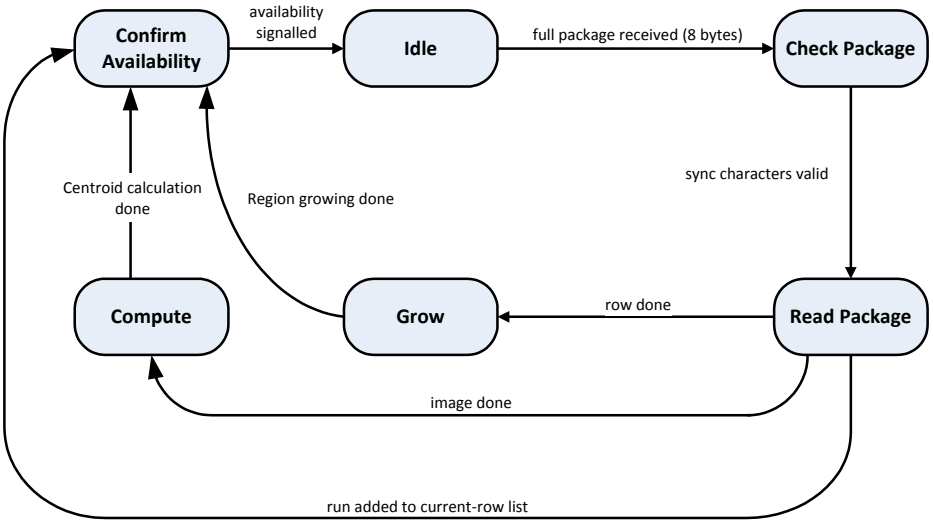


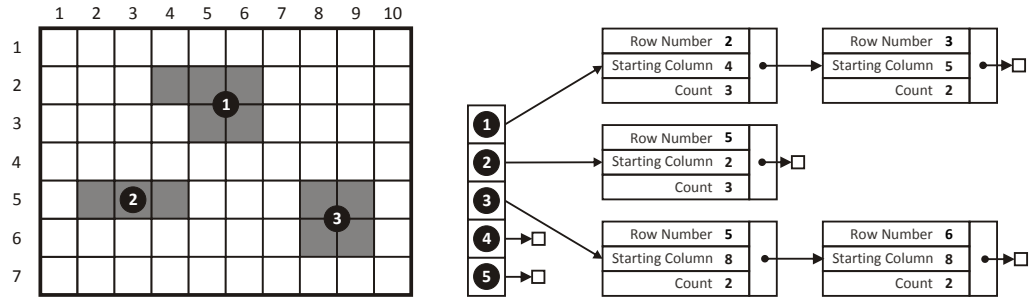
Figure 5.14: Simplified state diagram of package reception and region growing on the microcontroller

5.3.7 Region Growing

The region growing algorithm is responsible for reconstructing and detecting contiguous regions in the thresholded image in a pipelined fashion. The algorithm requires only the run data of the current and previous rows and is executed whenever the end-of-row sentinel is received from the FPGA. The contribution of the current row to the existing regions is determined by comparing it to the previous row and updating the appropriate regions based on the overlap the high-intensity regions in the current row has with the regions in the previous row. The regions are represented by a table of linked lists where each individual linked list contains all the runs that constitute a specific region. After the image has been fully received and the regions have been reconstructed, the centroid and mass of each region are calculated and added to a list which acts as the input to the matching algorithm.

Algorithm

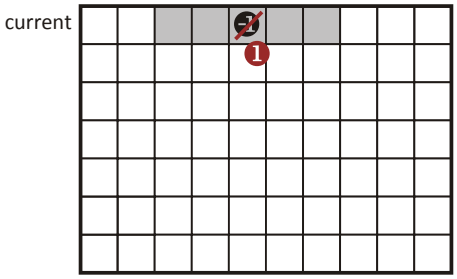
Each new run is assigned a sentinel identifier when it is added to the current-row linked list to indicate that its ownership by a region has not yet been determined (-1 in the examples of Figure 5.16). If a run has no overlap with a previous run, it is assigned a



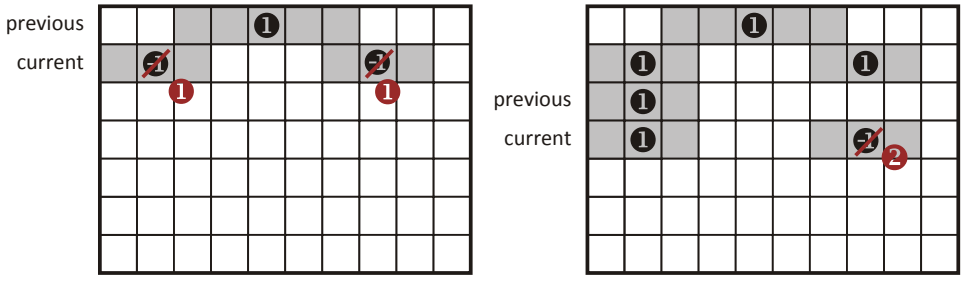
(a) Sample image with three regions (b) List representation of sample image

Figure 5.15: The representation of regions during the region growing process

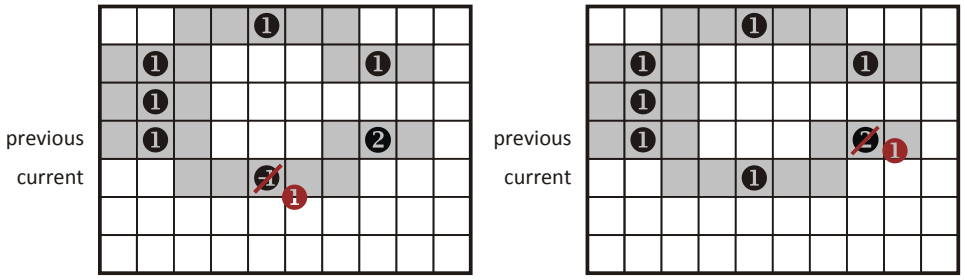
new identifier (Figures 5.16a and 5.16c) from a pool of available identifiers. If there is overlap between two runs in two consecutive rows, they form part of the same region and are assigned the same identifier (Figures 5.16b and 5.16d). A special case exists where the current run spans two regions in the previous row (Figure 5.16d). In this case the areas are merged and the identifiers are updated accordingly (Figure 5.16e).



(a) First Row, no overlap



(b) Both cases of simple overlap (c) No overlap, new region



(d) Special case, initial assignment (e) Special case, reassignment

Figure 5.16: The different overlap possibilities during region growing.

When the image is done and no more runs are available for processing, the centroid and mass for each region, m, is calculated [92] and stored in a linked list.

$$
\begin{aligned}
I_x^m &= \sum_{R^m_{\text{cols}}} \sum_{R^m_{\text{rows}}} p(i,j) \cdot i \\
I_y^m &= \sum_{R^m_{\text{cols}}} \sum_{R^m_{\text{rows}}} p(i,j) \cdot j \\
M &= \sum_{R^m_{\text{cols}}} \sum_{R^m_{\text{rows}}} p(i,j) \\
p_x^m &= \frac{I_x^m}{M^m} \qquad (5.1) \\
p_y^m &= \frac{I_y^m}{M^m} \qquad (5.2)
\end{aligned}
$$

Where the position of each region, m, is described by R^m. Regions with unrealistically small mass values (typically due to sensor noise) are excluded from the list before even being passed to the matching algorithm.

Implementation

The region growing algorithm is implemented using an array of linked-lists (Figure 5.15). Each element of the array represents a region and contains a linked list of runs that form part of that region. When a new region is started the list is added to the first available array element (identified by a NULL pointer). When an overlapping run has been found it is added to the appropriate list. When the special case occurs, the list that contain the runs for which the identifier must be updated are simply appended to the new identifier's list while the old identifier is marked as available again. After the entire image has been processed by the region growing algorithm, the centroid position and mass for each region is calculated (using Equations 5.2 and 5.2). The result is stored in another linked list. Linked lists are preferred to static arrays in the algorithms used on the microcontroller to allow a more robust implementation and allow the same low level firmware to be used for different applications with different marker configurations without sacrificing memory or requiring firmware changes. At this stage the original region information is not required and the allocated memory is freed and replaced on the heap.

5.3.8 Matching and Pose Estimation

The input to the matching algorithm is the linked list containing the centroid and mass of each high intensity region found in the image. The state information required to generate the virtual photograph used in the optimisation is placed on the CAN bus by the OBC at 50 Hz. Right before an image is captured the last state values are buffered and stored for the matching computation. This is done to ensure the state estimates resemble the true states when the image was captured. The matching is performed by an implementation of the algorithm developed in Section 2.5. The result of the matching algorithm is another linked list that contains only five elements – the centroid and mass of the best candidate

Table 5.5: CAN packets transmitted from the camera node

ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0C800901	North Measurement		East Measurement		Down Measurement		Status 1	
0C810901	Roll Measurement		Pitch Measurement		Yaw Measurement		Status 2	
0C820901	$C_{1,x}$	$C_{1,y}$	$C_{2,x}$	$C_{2,y}$	$C_{3,x}$	$C_{3,y}$	$C_{4,x}$	$C_{4,y}$
0C830901	$C_{5,x}$	$C_{5,y}$						

Table 5.6: CAN packets received from the OBC

ID	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0C900901	Integration Time		Gain Setting		Compression Threshold		Matching Threshold	
0C910901	North Estimate		East Estimate		Down Estimate			
0C920901	Roll Estimate		Pitch Estimate		Yaw Estimate			

marker projections. This list is used as the input to the implementation of the pose estimation algorithm developed in Section 2.6. After the computation the estimated visual state vector of the vehicle is returned to the OBC in two CAN packets with status information about the measurement cycle. In addition, the five chosen centroids are returned as well for logging and analysis. The packages are summarised in Tables 5.5 and 5.6.

5.4 Summary

In this chapter the practical implementation of the camera node was discussed. It outlined decisions made with regards to component choices in Section 5.2. The algorithmic flow was outlined in Section 5.3 and implementation details for each major module in the system were discussed. This ranged from timing details of the image sensor to custom communication protocols between the hardware components.

Chapter 6

System Simulation

6.1 Introduction

Simulation is an important step in the design process used to verify the functionality and implementation of the control and estimation algorithms as well as the avionics and other related systems. It relies on a non-linear mathematical model that mimics the behaviour of the helicopter to perform the validation. The model used during this project was developed by [31].

In this project two types of simulation environments were used: software (discussed in Section 6.2) and Hardware-in-the-Loop (HIL) (discussed Section 6.3). The modifications done to the existing hardware-in-the-loop system developed by [37] to support the vision system are also outlined in Section 6.3. Results of the simulations are shown and discussed in Section 6.4.

6.2 Software Simulation

Software simulations are perfomed in a software-only environment where there is no interaction with avionics or other flight related hardware. It is very convenient due to the amount and rate of data that can be logged for analysis as well as the very quick turnaround time for modifications. Since there is no interaction with external hardware, the simulation can be performed faster than real time. As a result, software simulations played a major role during the development of this project.

6.2.1 Autopilot System

The software simulation is implemented in the MATLAB/Simulink environment and contains all the autopilot components (estimator, outer loop algorithms, inner loop controllers), virtual sensors and non-linear model of the helicopter.

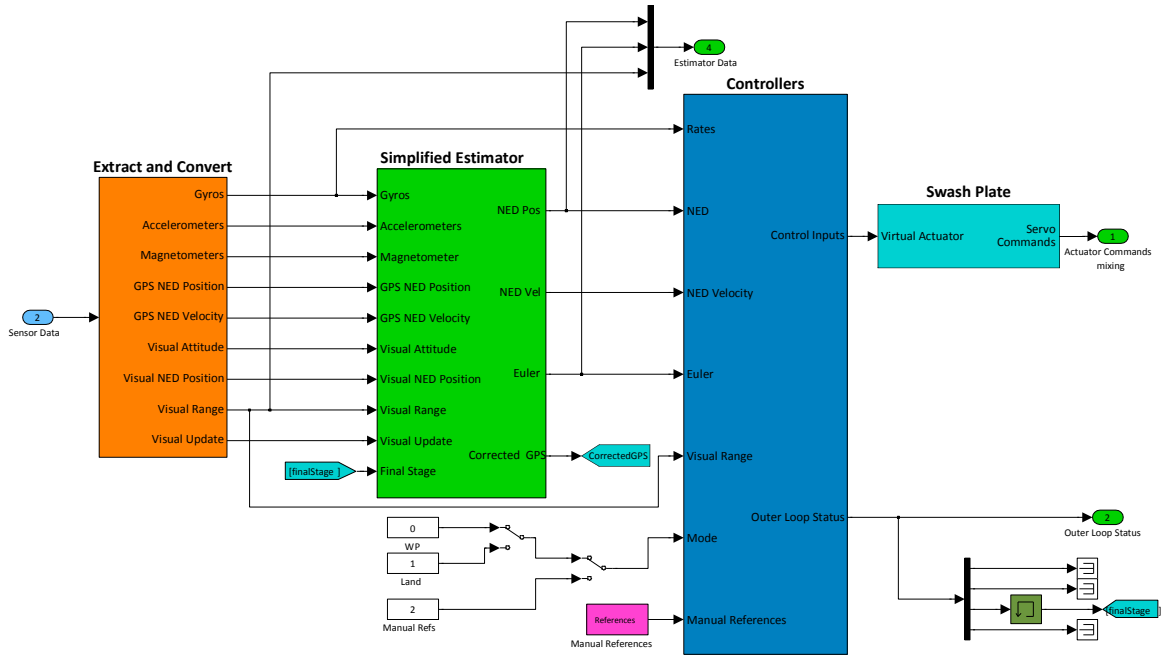


Figure 6.1: Autopilot overview in the software simulation

In previous projects [31, 69], the autopilot components were developed in either block diagram form or MATLAB specific M-code. In this project a different approach is taken with the software simulation: code that forms part of the autopilot is wrapped in MATLAB S-functions with the appropriate conversions between data types and structures used on the OBC. This allows a much simpler transfer of code between the simulation and hardware implementations. Although the previous approach tested the functionality of the design, it did not directly contribute to the reliability of the system when implemented on the OBC. In addition, changes made in the software environment had to be duplicated in the hardware implementation which eventually lead to the software simulation being neglected and most development being done directly in the hardware simulation environment. Although this is suitable for small continuous development, the advantages of software simulation with wrappers during larger projects are significant:

1. Fast and easy access to a large number of signals for debugging and analyis.
2. Isolation of hardware and software errors during the implementation on the OBC.
3. Continuous testing of code that will eventually be used in flight.

An overview of the autopilot system is shown in Figure 6.1. In this simulation diagram the sensor data that is created by the non-linear model is extracted from the bus and passed to the estimator. An initialisation function in the estimator wrapper is invoked that loads the measurements and settings into the OBC data structures after which the usual estimator update function is called. When the estimator has finished its update for the specific timestep, the results are extracted from the OBC data structures by the wrapper and loaded into the appropriate Simulink signals (rates, NED position, NED velocity and

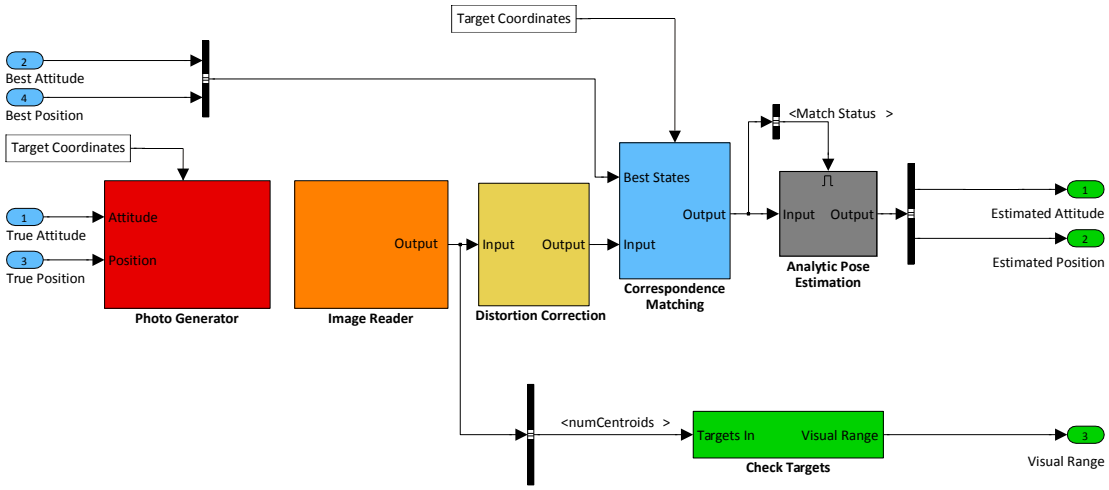


Figure 6.2: VPAM system in the software simulation

Euler angles). These signals are passed, with a number of commands and settings, to the controller module where similar initialisation functions in the inner loop and outer loop wrappers load the measurements and settings into the OBC data structures. Once loaded, the outer loop update is performed to generate position, velocity and heading references for the inner loop controllers. These measurements are extracted and loaded into the appropriate Simulink signals before they are loaded into the OBC structure by the inner loop's initialisation wrapper function. After the servo outputs have been determined by the inner loop controllers, they are extracted and passed to the non-linear helicopter model to create the forces and moments from which the next set of sensor measurements are created.

The flight mode is determined by a selection of switches (shown at the bottom of Figure 6.1): waypoint navigation, landing or manual references. Settings applicable to the outer loop controller and estimater are configured through settings dialogues in Simulink.

6.2.2 Vision System

The complete vision system was tested in simulation including the compression, region growing, matching and pose estimation. The simulation diagram is shown in Figure 6.2.

The photo generator block uses the true states and marker locations in the pinhole camera model (Section 2.3) to generate a virtual image of the landing target. This image is compressed and stored on the hard drive. The image reader opens the file and performs the region growing algorithm (Section 5.3.7) resulting in a list of centroids which is first corrected for barrel distortion before it is passed to the correspondence matching module (Section 2.5). The list of candidate target markers is passed to the pose estimation module (Section 2.6) where the relative attitude and position states are determined. All the vision processing algorithms (region growing, correspondence matching and pose estimation) were wrapped in the same manner as the estimator and controllers which simplified the hardware implementation.

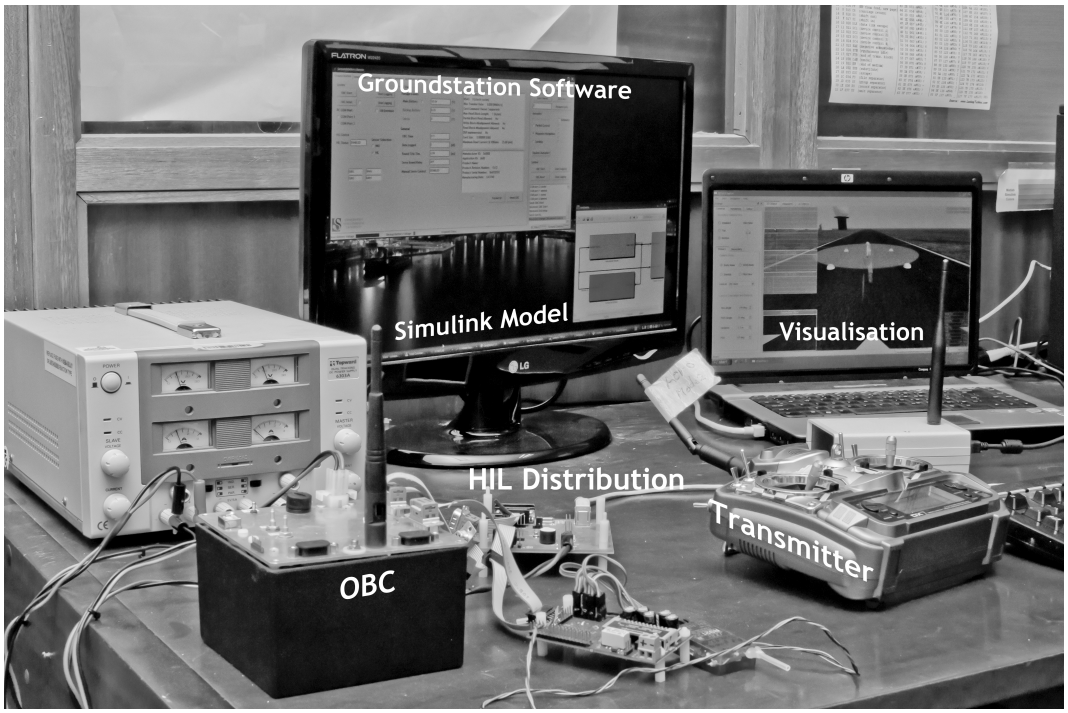


Figure 6.3: The HIL system interfaced to the avionics and computer

6.3 Hardware-in-the-Loop Simulation

A hardware-in-the-loop system is used in the ESL and it allows the estimator, controllers, avionics and associated subsystems to be tested in a controlled environment [37]. The system consists of a near real-time Simulink program that simulates the vehicle's behaviour through a non-linear model. Communication between the avionics and the Simulink program is done through the HIL distribution board and the USB interface. The setup of the HIL system is shown in Figure 6.3.

The non-linear vehicle model takes the control inputs and propagates the vehicle dynamics by 20 ms to calculate the resulting forces and moments acting on the vehicle. These forces and moments are transformed to the vehicle position, velocity and attitude states by a six-degree-of-freedom kinematic model. These states are converted into GPS, accelerometer, gyroscope and magnetometer measurements and transferred to the HIL distribution board via the USB interface where it is packaged into CAN messages. When the OBC issues a sync pulse (which would normally be received by the sensor nodes), the distribution board sends the virtual measurements (packaged to appear as if they originate on the sensor node) to the OBC. The estimation and control algorithms are executed, and the control signals, destined for the servo interface, are intercepted by the HIL distribution board and transferred to the PC for the next iteration. The system is explained in detail in [37].

6.3.1 Vision System

The process described in Section 6.3 is scheduled by the synchronisation pulses generated by the OBC. If the transfers and calculations take longer than the period between synchronisation pulses (20 ms) the system will not behave as intended. Since Microsoft Windows is not a real time operating system, there is a time delay between the control signal transmission from the HIL distribution board and the model propagation in Simulink. Since this delay cannot really be controlled[1], the amount of processing done in Simulink and the transmission time to, and from, the HIL distribution board must be kept to a minimum.

The baud rate between the PC and the HIL distribution board is 115 200 bps which results in the 112 bytes being transferred in 8.33 ms every cycle. This leaves only 11.67 ms to execute the propagation and kinematic equations and to create the measurements. When running the Simulink environment, groundstation software and OpenGL visualization environment, timeouts still occur, even on a powerful computer[2]. Further additions to the current system, especially in the Simulink environment, would not be ideal. The following actions were taken to accomodate the visual system in the HIL environment:

1. Since the VPAM node is independent of the OBC with regards to processing (unlike the first implementation [92]), it can be thoroughly tested through the software simulation described in Section 6.2.2 and standalone practical tests. Once the VPAM node's functionality has been proven, the interface to the node and other related systems (estimator, telemetry data link and data logging) can be tested in HIL without the VPAM node. This is achieved by generating visual measurements in Simulink by adding noise and a 100 ms delay to the true states when the markers are in the field of view.

2. The HIL distribution board and Simulink interface were modified to accomodate 16 additional bytes of data for transferring the visual measurements and associated status information. More data transferred from the HIL distribution board to the PC reduces the amount of time available for processing. To alleviate this problem, the baud rate between the HIL distribution board and PC was increased to 230 400 bps. This reduced the transfer time (including the visual measurements) to 4.86 ms, which reduced, but did not completely eliminate, the number of timeouts.

3. The HIL system has legacy support for the previous avionics architecture used in the ESL which used an analog interface instead of the CAN bus [63, 68]. The HIL distribution board performed DAC operations to generate the analog signals every cycle, even when using the CAN-based avionics. To increase resources and reduce delays on the distribution board, this support was removed as the CAN system has become the standard system used in ESL projects. An overview of the modified HIL system is shown in Figure 6.4.

[1] It can be minimised by reducing the number of applications running during HIL simulations.
[2] An Intel Core 2 Duo 3 GHz with 3 GB memory was used during this project.

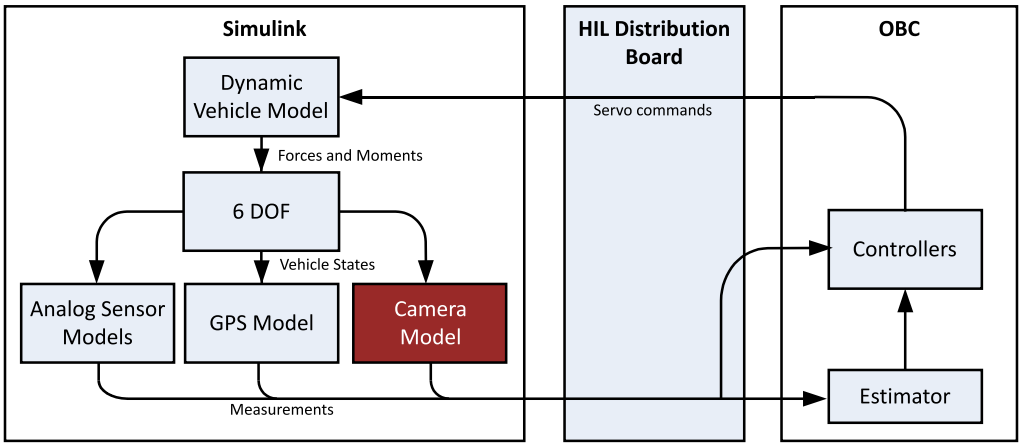


Figure 6.4: Modified HIL System

6.4 Results

Simulation results for the waypoint navigation and autonomous landing system are presented in this section. The scenario used in this simulation encapsulates both the waypoint navigation and autonomous landing systems while using the the visually augmented estimates when they are available. The first part of the simulation (Section 6.4.2) tests waypoint navigation on a circuit defined by four waypoints. After two complete circuits, the landing command is issued which guides the helicopter to the hover point above the landing target and eventually to the ground.

6.4.1 Waypoint Navigation

A 3-D view of the flight path is shown in Figure 6.5 which verifies the basic functionality of the guidance system. The controller parameters used in the waypoint navigation are shown in Table 6.1.

Table 6.1: Controller parameters for waypoint navigation

Quantity	Value	Description
$V^n_{X,\text{ref}}$	1 m.s^{-1}	Longitudinal Velocity Reference
ϵ^n_{pos}	1 m	Distance Threshold
ϵ^n_{ψ}	10°	Heading Threshold
τ^n_{hover}	2 s	Hover Time

The position, velocity and attitude states during the navigation procedure are shown in Appendix C.

6.4.2 Landing Procedure

A 3-D view of the landing procedure is shown in Figure 6.6 which verifies the basic functionality of the landing system. The position states are shown in Figure 6.7. The

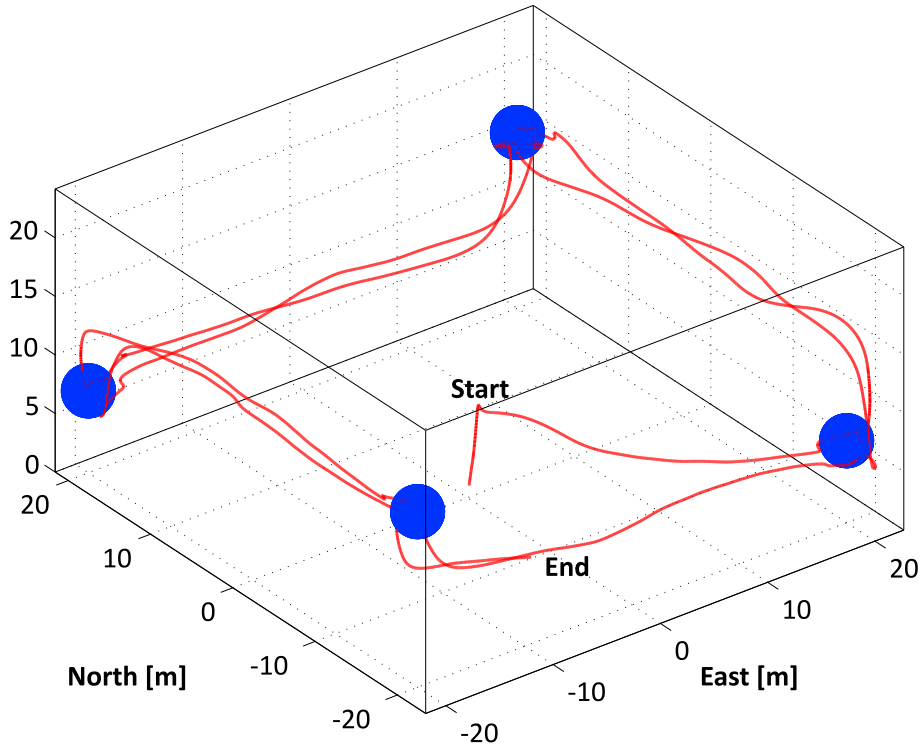


Figure 6.5: 3-D view of the navigation simulation

controller parameters for the landing procedure are shown in Table 6.2. The remaining states are shown in Appendix C.

Table 6.2: Controller parameters for autonomous landing

Quantity	Value	Description
Ψ^l_{ref}	45°	Heading reference during line up and descent
$v^l_{D,\text{ref}}$	0.3 m.s^{-1}	Descent Velocity Reference
$v^l_{X,\text{ref}}$	1 m.s^{-1}	Longitudinal Velocity Reference
ϵ^l_{pos}	1 m	Distance Threshold
ϵ^l_{ψ}	10°	Heading Threshold
$\mathbf{P}^l_{\text{hover}}$	[0 0 -8 m]	Hover Position
τ^l_{hover}	2 s	Hover Time

From the position states in Figure 6.7 it can be seen that the GPS and inertial sensors are used to guide the helicopter towards the hover point. When visual lock is obtained the smooth transition algorithm delays the immediate fusion with the current state estimates to prevent steps in the state estimate. Once the transition is complete, the visual measurements are fully combined with the inertial measurements to reduce the errors on the state estimates. When visual lock is lost, GPS is used again to perform measurement updates. Due to relative inaccuracy of the GPS module the estimates become less accurate, resulting in a reduction in touchdown accuracy.

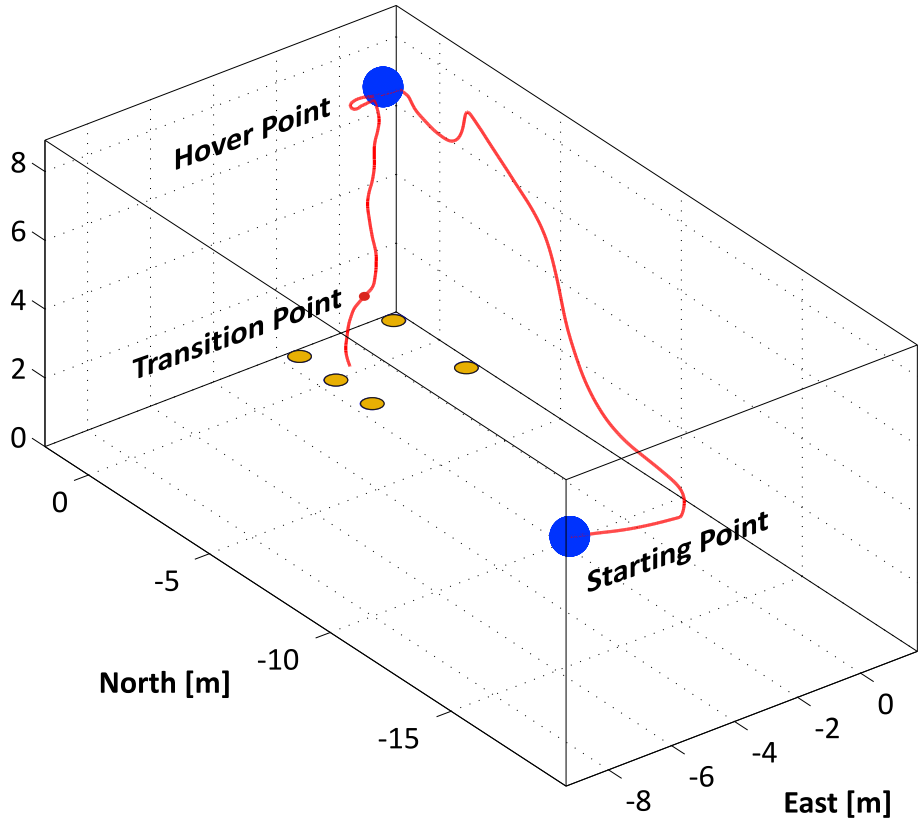


Figure 6.6: 3-D view of the landing simulation

6.4.3 Landing Accuracy

The aim of the ATOL landing project is to land within a circular area with a 1 m radius. This simulation performs a number of landings to determine the frequency with which this aim is achieved. The simulation performs waypoint navigation and (at a randomly selected time) transitions to the landing procedure. A history of the touch down point and the accuracy at the point where visual lock is lost (transition point) is maintained for the sequence of landings. Since the separation of markers in the landing target determines at what altitude the centroids will disappear from the field of view, the simulation is repeated a number of times for various separation distances to determine the effect the unaugmented estimates of the final part of the descent phase have on the accuracy.

The accuracy at touchdown and the accuracy of the transition point are shown in Figures 6.8a and 6.8b while a histogram of the radial accuracy is shown in Figures 6.8c and 6.8d. 48.7% of the touchdowns were within the 1 m region, while 48.6% was within a 2 m region. Only 2.7% was outside this region. At the transition point, the helicopter was within the 1 m region 94.9% of the time. The deviation between accuracy at the transition point and at touchdown could be attributed to the inaccurate sensors used for the final part of the descent after vision lock has been lost at 2 m altitude. The marker separation in this case is 100 cm between vertices of the landing target.

The obvious solution to improve the touchdown performance (given the very good tran-

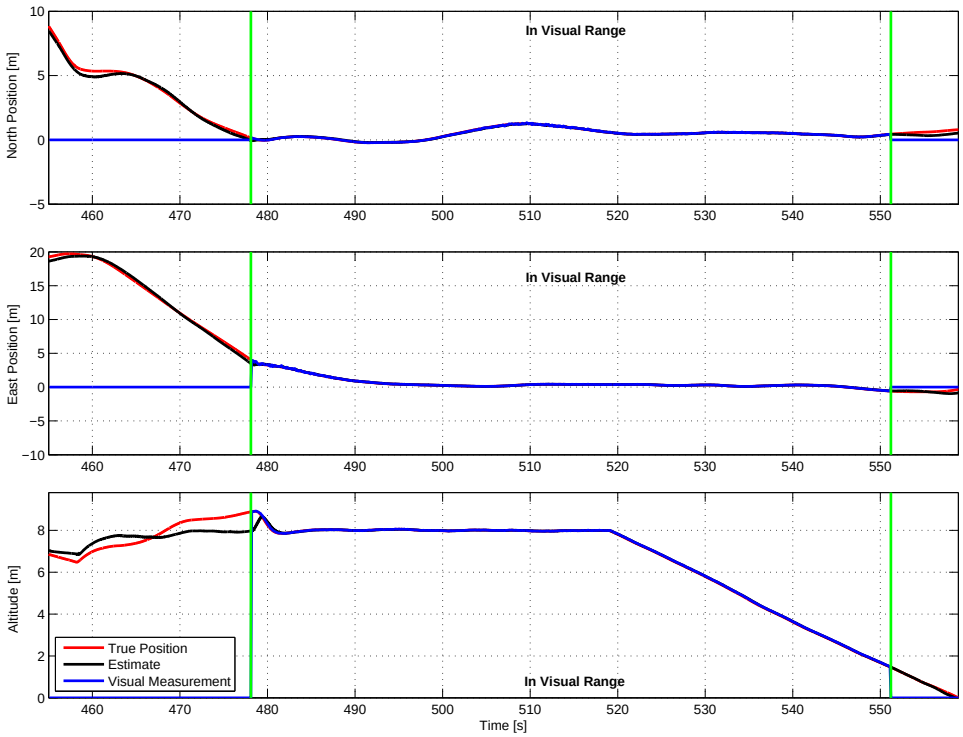
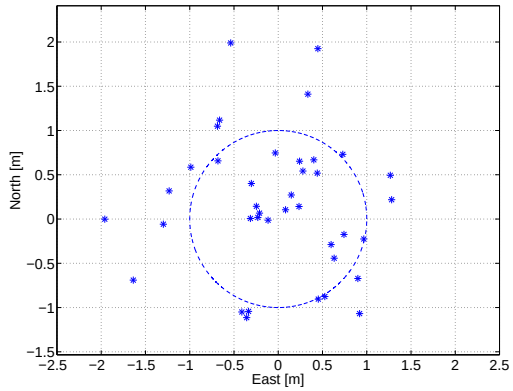


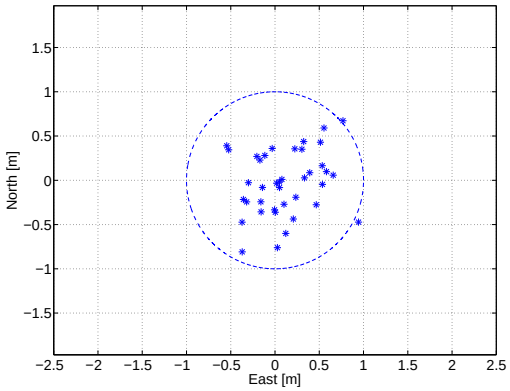
Figure 6.7: Position states during landing procedure

sition point accuracy) is to move the markers closer to each other to reduce the altitude of the transition point. This will, however, reduce the altitude at which visual lock can be established. Since accuracy at higher altitudes is less critical, GPS can be used for a longer period and the hover point altitude can be reduced to a point where visual lock is possible. The accuracy at touchdown and the transition point for a reduced marker separation of 56 cm is shown in Figures 6.9a and 6.9b while a histogram of the radial accuracy is shown in Figures 6.9c and 6.9d. 66.7% of the touchdowns were within the 1 m region, while 31.4% were within a 2 m region. Only 1.9% was outside this region. At the transition point, the helicopter was within the 1 m region 96.1% of the time. In this case the transition point is around 0.9 m which reduces the distance in which inaccurate measurements can cause significant deviation in the touchdown accuracy. At the transition point for the two separations the accuracy remains almost unchanged.

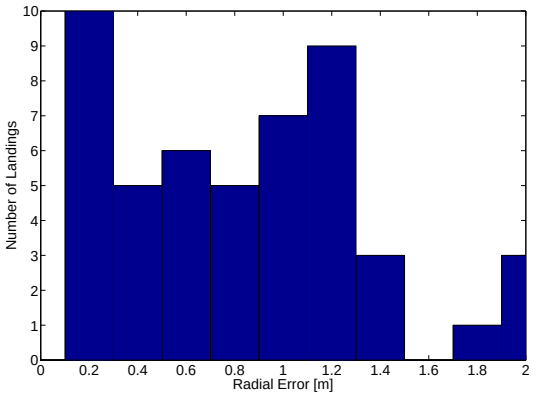
The touchdown accuracy for a number of separation distances is shown in Figure 6.10a while the accuracy at the transition point is shown in Figure 6.10b. The effect the low accuracy GPS and inertial sensors have on the touchdown accuracy is clear from the decrease in accuracy as the duration of the final phase of the descent increases. The mean radial error at the transition point is 40 cm which is comparable to the landing accuracy obtained in [72] using a significantly more expensive system. To achieve similar landing performance, a better measurement strategy for the final part of the descent must be investigated in future projects. A very plausible solution is to have a second set of markers (with much closer spacing than the original set) that can be used to provide



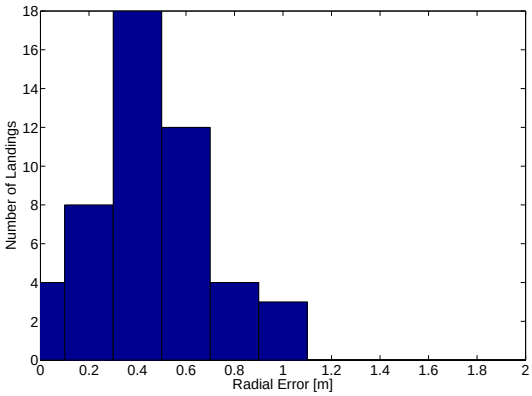
(a) Touchdown point accuracy



(b) Transition point accuracy



(c) Radial error distribution for touchdown point



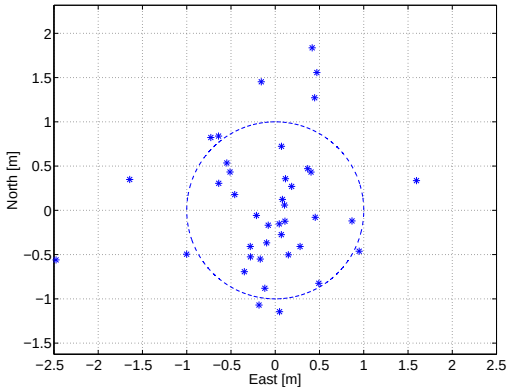
(d) Radial error distribution for transition point

Figure 6.8: Accuracy during descent stage for 100 cm separation

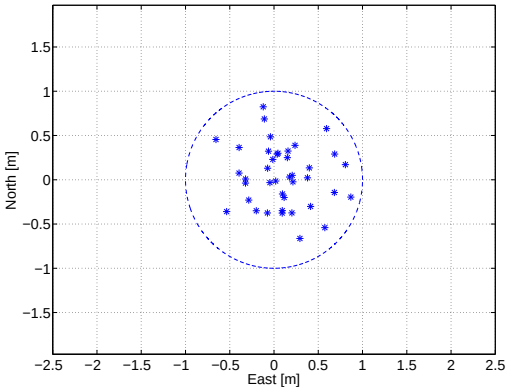
vision-based measurements when the original set leaves the field of view at lower altitudes. Figure 6.10a can be used as a guideline in the design of the marker system to obtain a specific level of touchdown accuracy.

6.5 Summary

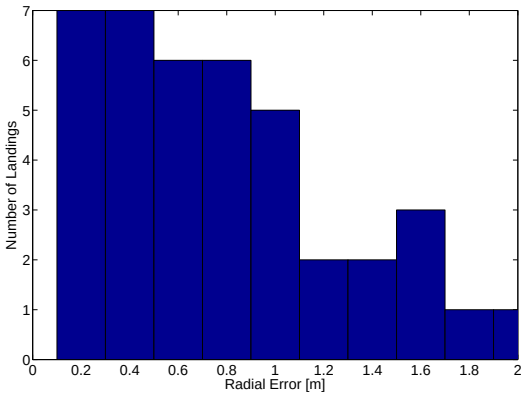
In this chapter the two simulation methods used during this project were shown. Changes made to the HIL system to accomodate the visual system were motivated. A series of simulations were performed to confirm the functionality of the vision system, navigation controller, landing controller, estimator and inner loop controllers. Finally, a simulation was performed to show the effect of marker separation on the touchdown accuracy.



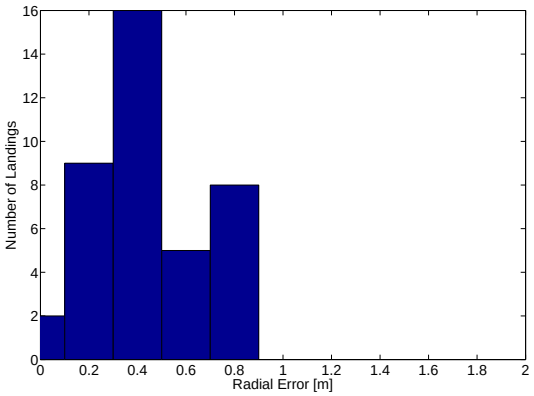
(a) Touchdown point accuracy



(b) Transition point accuracy

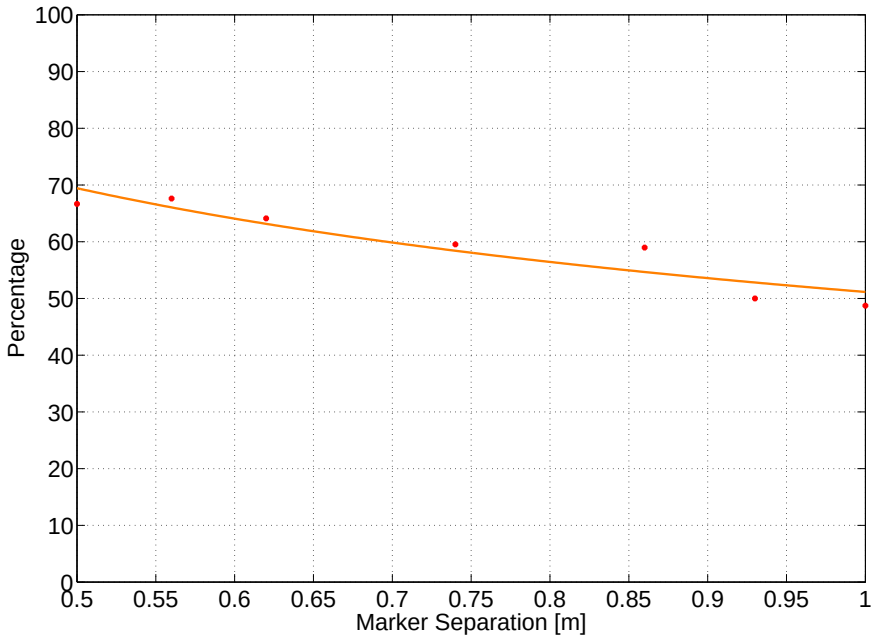


(c) Radial error distribution for touchdown point

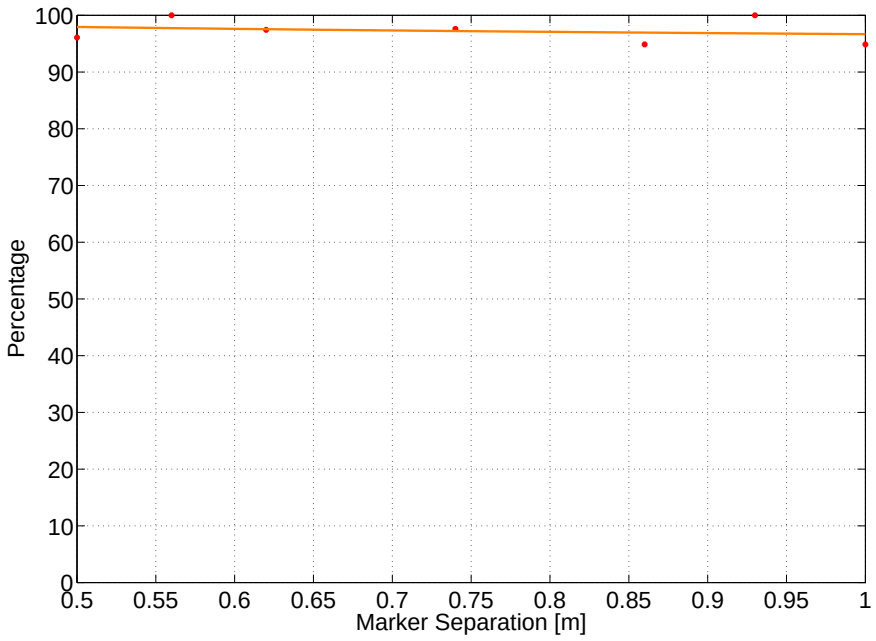


(d) Radial error distribution for transition point

Figure 6.9: Accuracy during descent stage for 56 cm separation



(a) Percentage of valid touchdowns versus marker separation



(b) Percentage of valid transition point measurements versus marker separation

Figure 6.10: Percentage of valid measurements and touchdowns versus marker separation

Chapter 7

Practical Considerations and Results

7.1 Introduction

This chapter provides an overview of some practical aspects of the VPAM node as well as practical results. The practical aspects include the camera enclosure (Section 7.2.1), mounting offset correction (Section 7.2.2), distortion correction (Section 7.2.3) and camera calibration (Section 7.3). Two test configurations to test the VPAM node's functionality and accuracy are outlined in Section 7.4. Practical results are also presented.

7.2 Camera Design

7.2.1 Enclosure

Since the camera relies on the assumption that only infrared light (from the markers) will reach the image sensor, a narrow-band interference filter is used in front of the lens. This will prevent sunlight (or other artifical light) from reaching the image sensor through the lens. The inside of the enclosure is painted matt black to reduce reflections if stray light were to enter through openings in the enclosure.

An aluminium enclosure is used for protection against debris and impact since the camera will be mounted at the bottom of the helicopter. The camera is shown in Figure 7.1.

7.2.2 Mounting Offset

The position- and attitude-measurement algorithms derived in Chapter 2 determine the position and attitude of the camera relative to the landing target centred at the inertial frame's origin. Since the position and attitude of the helicopter is the required measurement, the mounting offsets (position and attitude) of the camera's optical centre relative to the helicopter's centre of gravity must be considered. The helicopter's position and attitude is determined through Equations 7.1 and 7.2:

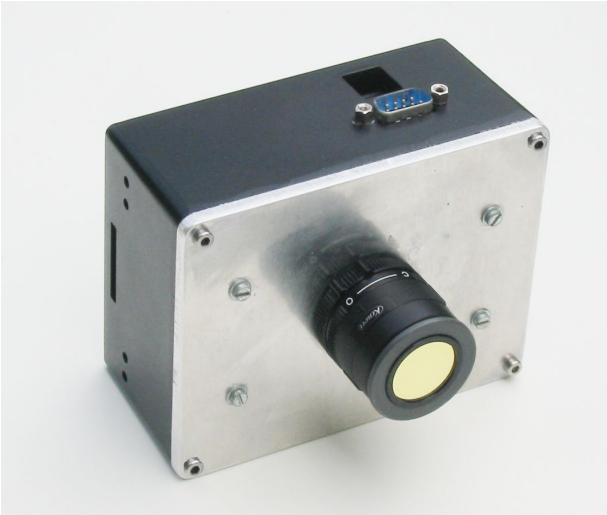


Figure 7.1: The VPAM node

$$\mathbf{P}^{I}_{\text{heli}} = \mathbf{P}^{I}_{\text{cam}} - \mathbf{DCM}_{B\to I} \cdot \mathbf{P}^{B}_{\text{cam}\to\text{heli}} \tag{7.1}$$

$$\mathbf{E}^{I}_{\text{heli}} = \mathbf{E}^{I}_{\text{cam}} - \mathbf{DCM}_{B\to I} \cdot \mathbf{E}^{B}_{\text{cam}\to\text{heli}} \tag{7.2}$$

where $\mathbf{P}^{B}_{\text{cam}\to\text{heli}}$ is the position of the camera's optical centre relative to the helicopter's centre of gravity coordinated in the body frame and $\mathbf{E}^{B}_{\text{cam}\to\text{heli}}$ is the camera frame's orientation relative to the body frame coordinated in the body frame.

7.2.3 Radial Distortion

Radial distortion is a non-ideal property of lenses that results in a deviation from rectilinear projection. This results in the projections of straight lines becoming curved. Since rectilinear projection (where the projections of straight lines remain straight) is assumed for the algorithms developed in Chapter 2 (through the use of an ideal pinhole model) the distortion must be corrected before the image can be used for measurements. The simplest method to compensate for radial distortion is to use low-distortion lenses that have elements in the optical path to reduce the distortion. These are very expensive and do not fit in the low-cost framework of the project. The alternative, and also the focus of many research projects, is to remove the distortion mathematically [26, 28, 84]. The most common method to remove the radial distortion is to model the distortion using a polynomial model [56]. This model is defined in Equation 7.3:

$$\begin{bmatrix} x_u \\ y_u \end{bmatrix} = \begin{bmatrix} x_{COD} \\ y_{COD} \end{bmatrix} + \begin{bmatrix} x_d - x_{COD} \\ y_d - y_{COD} \end{bmatrix} \cdot \left[1 + k_1 r_d^2 + k_2 r_d^4 + \ldots + k_p r_d^{2n}\right] \tag{7.3}$$

where (x_u, y_u) is the undistorted point and r_d is the distortion radius defined from the center of distortion to the distorted point (x_d, y_d):

$$r_d = \sqrt{(x_d - x_{COD})^2 + (y_d - y_{COD})^2}$$

This model can be simplified by truncating the infinite power series to include only the first two power terms and by assuming the centre of distortion coincides with the principle point (pp_x, pp_y) [56]:

$$\begin{bmatrix} x_u \\ y_u \end{bmatrix} = \begin{bmatrix} pp_x \\ pp_y \end{bmatrix} + \begin{bmatrix} x_d - pp_x \\ y_d - pp_y \end{bmatrix} \cdot \left[1 + k_1 r_d^2 + k_2 r_d^4\right] \tag{7.4}$$

with the distortion radius r_d given by:

$$r_d = \sqrt{(x_d - pp_x)^2 + (y_d - pp_y)^2}$$

Barrel-distortion correction is usually performed on an entire image, since the entire image is used by the low-level algorithms in typical computer vision applications. This is computationally expensive and unnessary for this project since only a small selection of points are used to determine the attitude. As a result only the centroids of all high intensity regions are corrected using Equation 7.4 before the correspondence matching algorithm is applied. The values of the principle point and the distortion model parameters (k_1, k_2) are determined through a camera calibration procedure outlined in Section 7.3.

7.3 Calibration

Camera calibration is the process through which the intrinsic parameters (f, pp_x, pp_y), relative pose between the camera and a calibration target and distortion model parameters (k_1, k_2) of the camera are determined. This is also a very active field of research within computer vision [67, 100, 34, 84, 90] and Zhang's method [100] forms the base of many of these algorithms. It is an advanced method that requires multiple photographs of a test pattern, but it is capable of determining the intrinsic, extrinsic and distortion model parameters without any prior knowledge of the camera parameters. Since some parameters of custom developed cameras are known within certain accuracy, this method was simplified to only require a single photograph of the calibration target [56]. The development of the simplified method is described in detail in [56], but a brief overview is given below:

1. Mount the camera perpendicular to a test pattern in a custom-designed clamp.
2. Position the camera such that the test pattern fills the entire frame.
3. Align the camera's optical axis to be perpendicular to the the test pattern.

4. Measure the test pattern's rotation and translation in the camera frame.

5. Take a full resolution image and extract the centroids.

6. Use a non-linear minimisation algorithm to determine the camera parameters.

The suggested test pattern is a square grid of 121 points drawn on a whiteboard. Since the output of the VPAM node developed in Chapter 5 is a compressed image of high contrast scenes, such a test pattern will not result in an adequate image. In addition, the focal length of lenses varies with the wavelength of the light [21]. Performing the calibration in the infrared spectrum instead of the visible light spectrum would be ideal since the focal length would be more accurate.

The proposed solution is a square aluminium sheet with a grid of 121 holes placed in front of an infrared lamp instead of the whiteboard test pattern. To ensure even coverage of the entire grid, a sheet of diffusion material is placed between the grid and the lamp (Figure 7.2). Cardboard guards were placed on either side of the test pattern to prevent direct light from reaching the lens. Photographs of this configuration are shown in Appendix D.

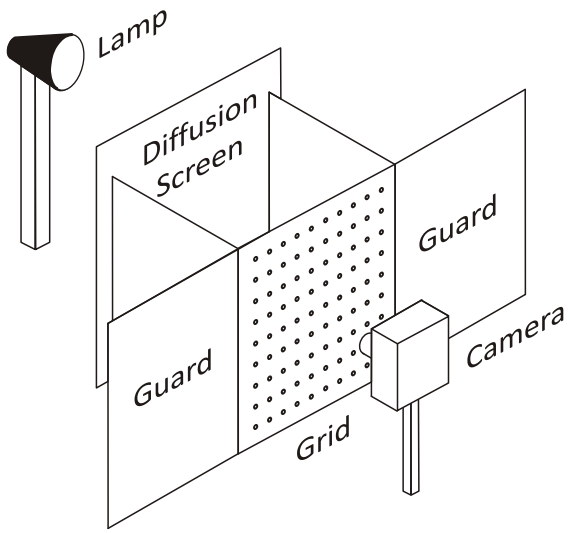


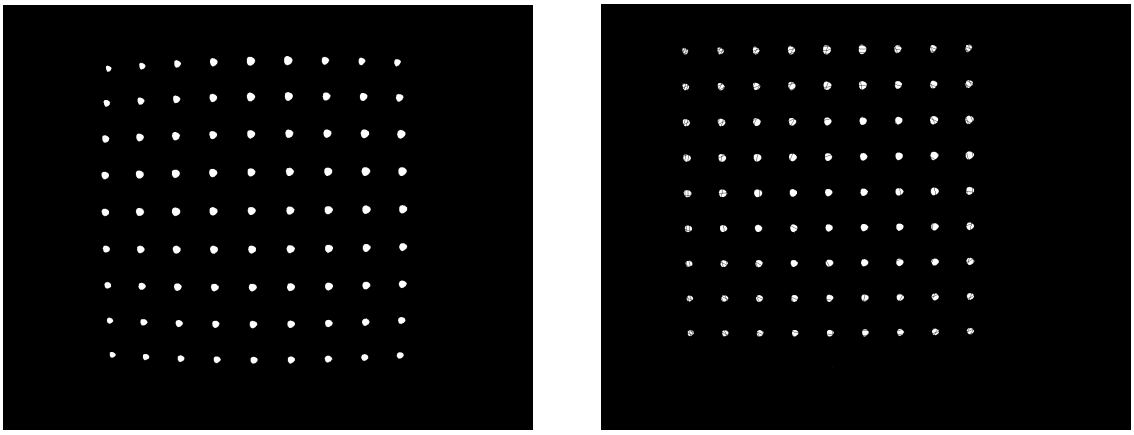
Figure 7.2: Calibration setup

During calibration, however, it was found that the projections of the extreme points in the grid were very small (or non-existant) and did not result in reliable centroids for the remainder of the calibration procedure. As a temporary solution, to get an approximation of the camera parameters, the interference filter was removed and visible light was used for the calibration. To prevent ambient light from influencing the exposure, the calibration was performed in a dark room. Unfortunately, due to the small size of the holes, the most extreme grid points were still not well formed and resulted in only the inner most 9 rows and columns of the grid being used in the minimisation process. Since barrel distortion is most pronounced at the edges of the frame, the accuracy of the distortion correction for images covering the frame is expected to not be optimal. This can be solved using a

calibration plate with larger holes to allow more light to pass, even at the extreme points of the grid. Unfortunately, time did not allow for a new plate to be manufactured and the tests were performed using the parameters shown in Table 7.1 obtained using the 9 by 9 grid. The result of the barrel-distortion correction is shown in Figure 7.3.

Table 7.1: Camera parameters

Parameter	**Value**
Focal length (f)	7.95 mm
Principle point (pp_x, pp_y)	$(663.63, 516.69)$
Distortion parameters (k_1, k_2)	$(2.4329 \times 10^{-7}, 8.8789 \times 10^{-14})$



(a) Uncorrected image (b) Corrected image

Figure 7.3: Barrel-distortion correction

The image sensor chosen in Section 5.2.1 has two processing pipelines: one for the odd rows in the array and one for the even rows. The two modules must be calibrated to reduce the offset between rows. Due to time constraints this calibration was not performed and the result is that a row could occasionally be below the threshold in the compressor. This leads to phantom centroids (shown in Figure 7.4), that will be handled correctly by the matching algorithm, but since it does result in a shifted centroid it reduces accuracy.

7.4 Results

This section outlines the results of two series of tests that were performed to confirm the functionality of the system. The first test is a small-scale test used to test the functionality during development and is discussed in Section 7.4.1. The second test, discussed in Section 7.4.2, is a full-scale test used to confirm the functionality as well as determine the accuracy of the system in a more practical situation. In the full-scale results presented in Section 7.4, trials that were affected by the phantom centroid problem mentioned in the previous section were repeated to remove spurious measurements. Since the aim of the small-scale

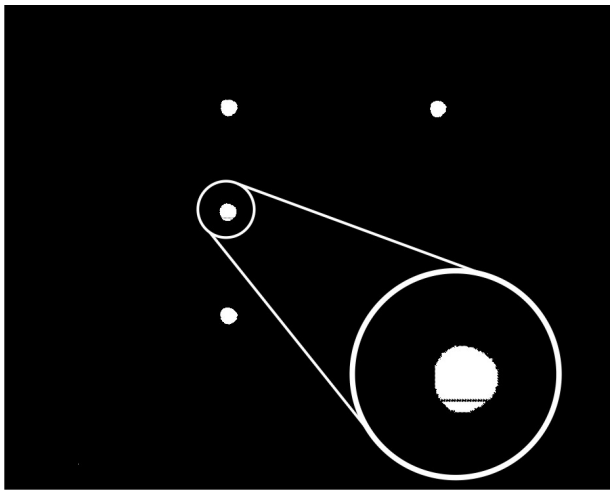


Figure 7.4: Phantom centroids due to uncalibrated ADC on the image sensor

experiments was simply to test the functionality of the VPAM node and not confirm the accuracy, this was not done for the tests in Section 7.4.1.

7.4.1 Small-Scale Test

The test rig developed to perform small-scale tests and algorithm verification is shown in Figure 7.5. High-power infrared LEDs were used as target markers with 11.8 cm separation between vertices. The distance between the camera and the markers can be adjusted to simulate altitude measurement while the markers can be moved sideways to simulation translation measurement. Photos of this test configuration are shown in Appendix D.

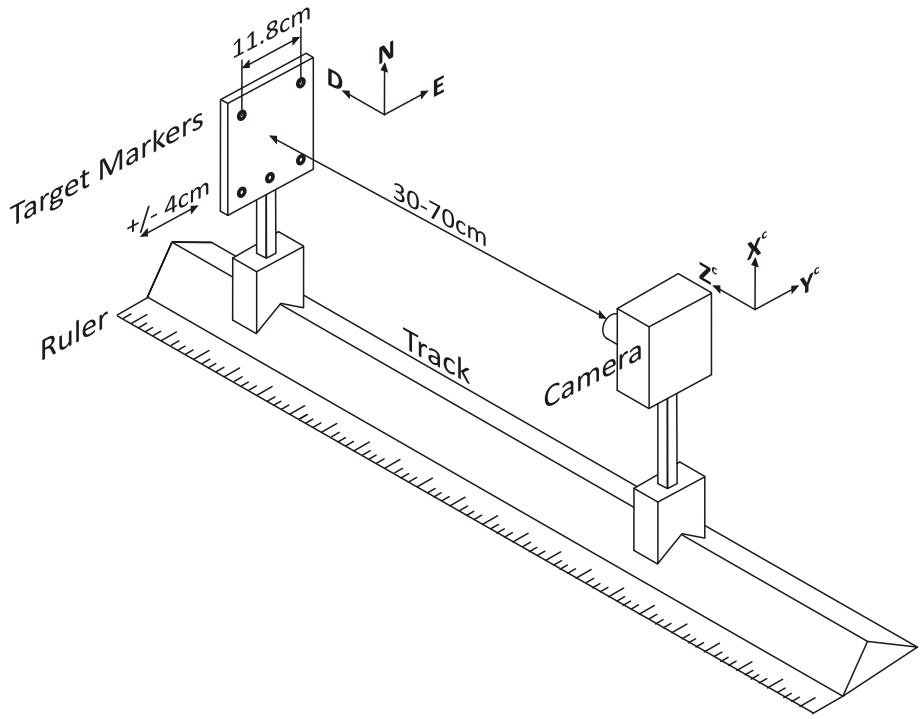
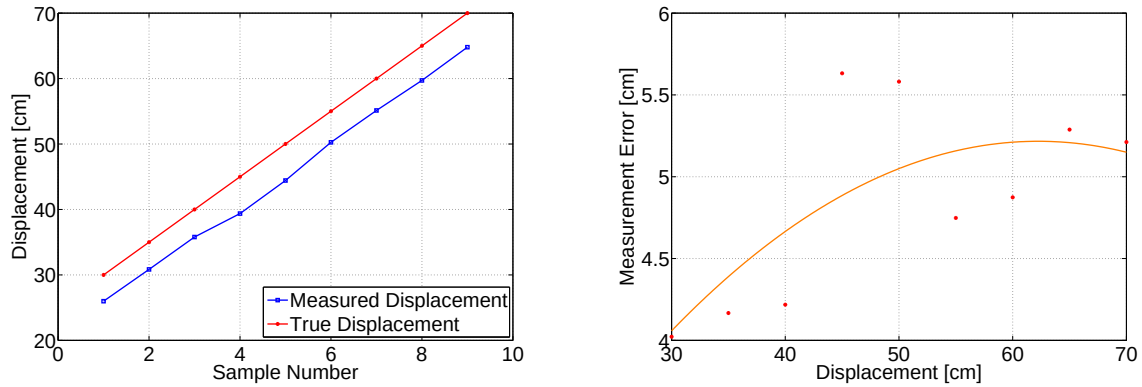


Figure 7.5: Small-scale test

Range Test

During this test the camera was centred with the target before the distance between the camera and the target was increased in a sequence of discrete steps. A comparison of the true distance and the measured range is shown in Figure 7.6a, while the measurement error is shown in Figure 7.6b.



(a) Range comparison

(b) Measurement error with parabolic fit

Figure 7.6: Small-scale test: Range results

The following observations are clear from Figure 7.6:

1. The basic functionality of the VPAM node with regards to range measurements has been tested by the small-scale tests. The different components (image readout, compression, region growing, correspondence matching and pose estimation) of the VPAM node all function together to provide attitude and position measurements.

2. There is an offset between the true distance and the measured value which can be attributed to non-circular shapes of the high intensity projections (Figure 7.8). This shape results in a bias on the calculated centroid which has a direct impact on the accuracy of the pose estimation.

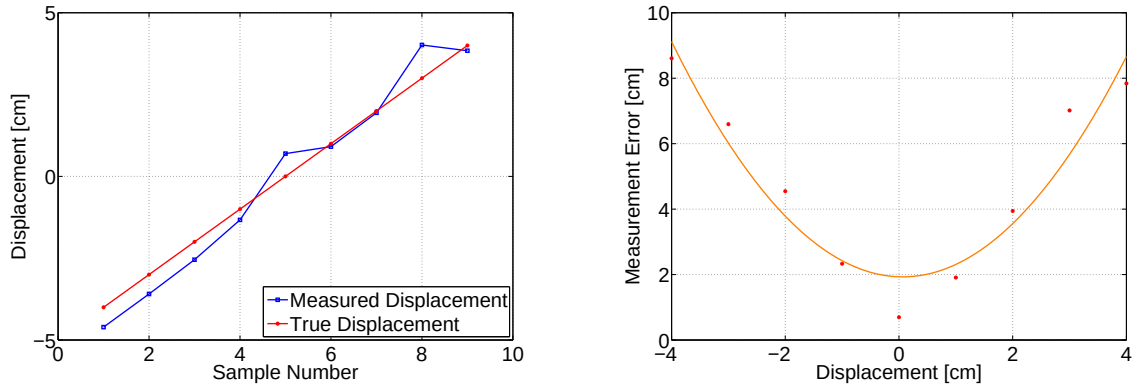
Translation Test

During this test the camera was centred with the landing target before the target was translated relative to the camera. The separation between the camera and the target was 30 cm. A comparison of the true distance and the measured translation is shown in Figure 7.7a, while the measurement error is shown in Figure 7.7b.

The following observations are clear from Figure 7.7:

1. The basic functionality of the VPAM node with regards to translation measurements has been tested through the small-scale tests.

2. The error between the true translation and the measurement increases with increasing translation distances due to the effect of inadequate barrel-distortion correction



(a) Translation comparison (b) Measurement error with parabolic fit

Figure 7.7: Small-scale test: Translation results

of projections at the edge of the frame and deformation of the projections due to the very narrow beam angle of the marker LEDs.

3. For improved performance with the high-efficiency LEDs, methods to improve the shape of the projections must be considered. These include mounts with reflectors, lenses and diffusion materials.

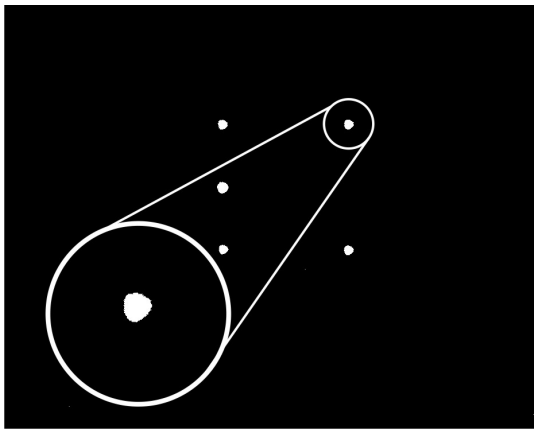


Figure 7.8: Non-circular shape of the high intensity projection

7.4.2 Full-Scale Test

The aim of the full-scale test is to confirm the functionality of the VPAM node in a more practical situation as well as investigate the measurement accuracy. The test involves a full-scale target mounted in an upright position with the camera mounted in a forwards-facing position on a trolley. By moving the trolley predetermined distances from the landing target the measurement accuracy can be determined. This configuration is illustrated in Figure 7.9 and photos are shown in Appendix D.

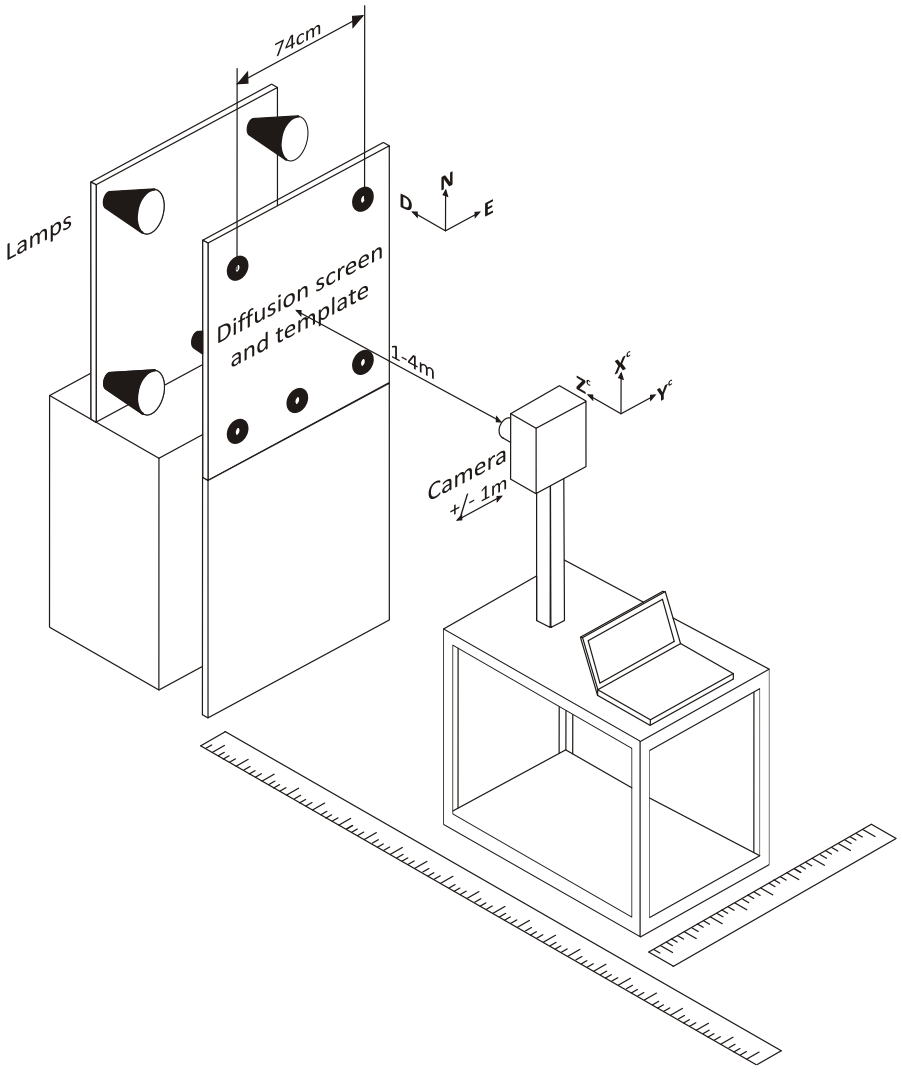
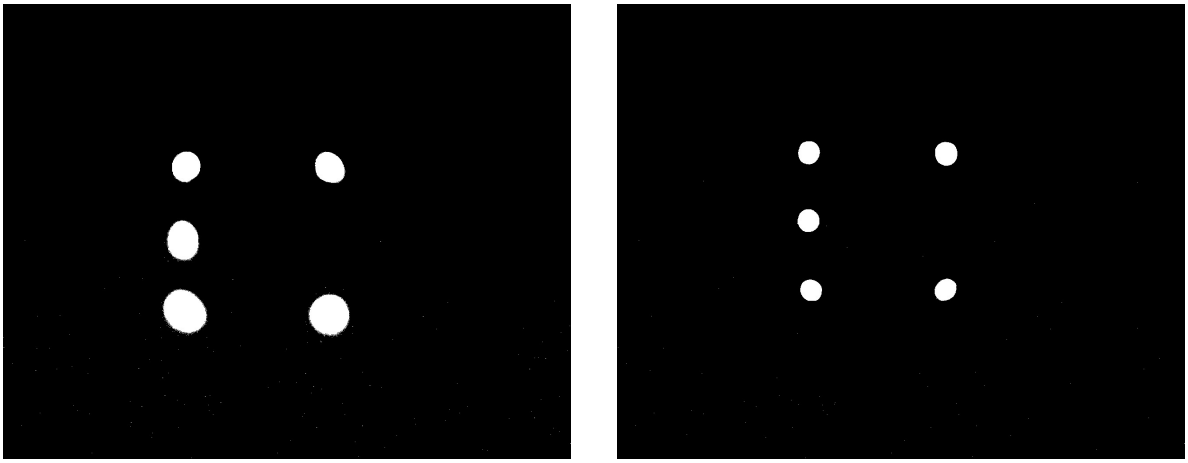


Figure 7.9: Full scale test

Five high-power lamps with significant power in the IR band (2.2 W per 10 nm – around 950 nm per lamp [92]) were mounted onto a vertical platform for this test. Initial tests resulted in low accuracy (±20 cm) measurements which were attributed to very large and non-circular projections. The size of the projections at close range, even with low gain and fast shutter speeds, resulted in a slower download time for each frame (since more rows had to be transferred to the microcontroller and processed). Despite claims in [14] that the image sensor has a synchronous shutter, a small amount of integration still takes place during the readout procedure. As a result of the longer download times, projections in the bottom of the frame experience a longer integration period which leads to the distortion shown in Figure 7.10a. This was solved by cutting smaller holes in a thin board where the markers should be, covering them with diffusion material and placing the board in front of the lamps. This reduced the size of the projections, which reduced the processing time and prevented the integration during readout from having a significant impact on the centroids. In addition, it improved the shape of the centroids causing them to be near circular as shown in Figure 7.10b.



(a) Distortion due to bare lamps (b) Diffused lamps

Figure 7.10: Diffusion of lamps in full-scale setup

Range Test

During this test the camera was centred with the target before the distance between the camera and the target was increased in a sequence of discrete steps between 2 m and 4 m. A comparison of the true distance and the measured range is shown in Figure 7.11a, while the measurement error is shown in Figure 7.11b.

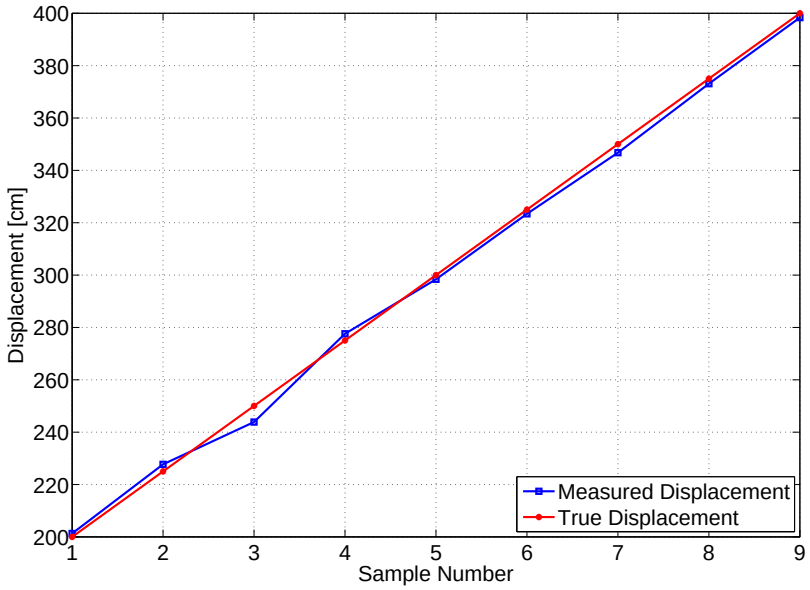
The following comments can be made regarding the results:

1. The VPAM node has the desired functionality with regards to altitude measurements in a full-scale setup.

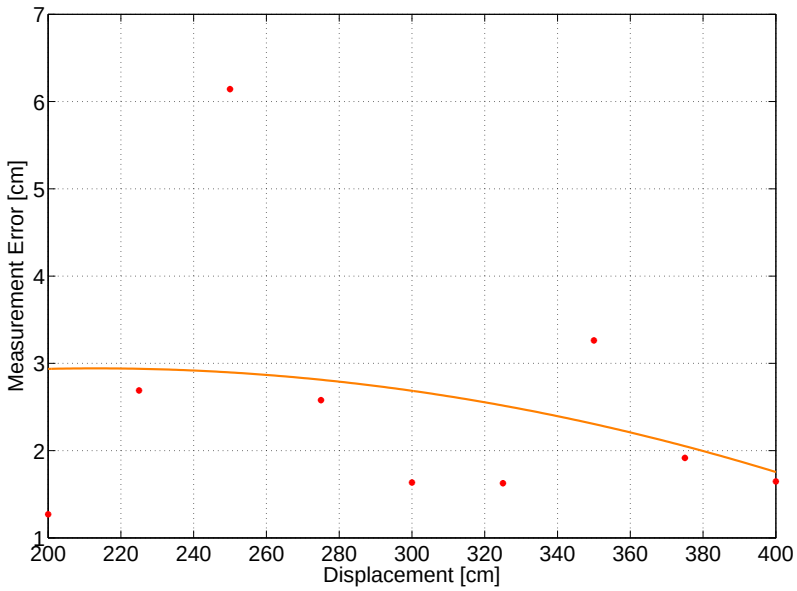
2. The error between the measured range and the true distance is very small – in the order of 3 cm which is similar to the accuracy obtained by other VPAM developments mentioned in Chapter 2.

3. The error decreases when range increases (Figure 7.11b) which is slightly counter-intuitive. This can be attributed to two factors: non-circular projections at close range and non-ideal barrel-distortion correction.

    - At close range, the projections are still relatively large and result in some degree of projection distortion due to increased integration time. This can be fixed by reducing the power of the target markers such that it is still detectable at the hover distance with higher image sensor gain, but does not result in over exposure (saturation) at a lower image sensor gain at the transition point.
    - At close range, the projections are close to the edge of the frame. Since the barrel distortion parameters obtained during the calibration procedure were not determined by a full-frame image, the barrel-distortion correction at the edge of the frame does not completely remove the distortion. This results in a reduction of pose estimation accuracy at close range. This effect can be reduced by a more accurate calibration procedure.



(a) Range comparison

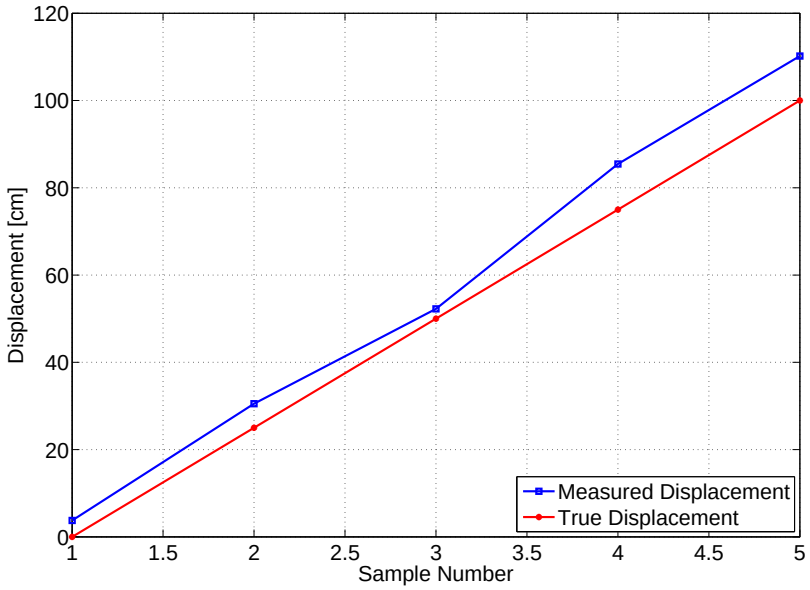


(b) Measurement error with parabolic fit

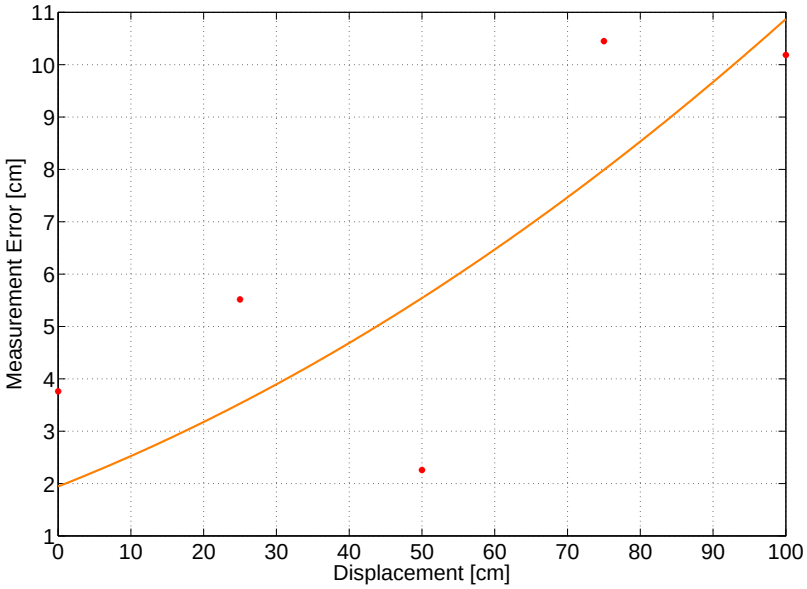
Figure 7.11: Full-scale test: Range results

Translation Test

As before: the camera was centred with the target before the camera was translated relative to the landing target in a number of discrete steps between 0 m and 1 m. The test was performed at a distance of 3 m from the target. A comparison of the true distance and the measured translation is shown in Figure 7.12a, while the measurement error is shown in Figure 7.12b.



(a) Translation comparison



(b) Measurement error with parabolic fit

Figure 7.12: Full-scale test: Translation results

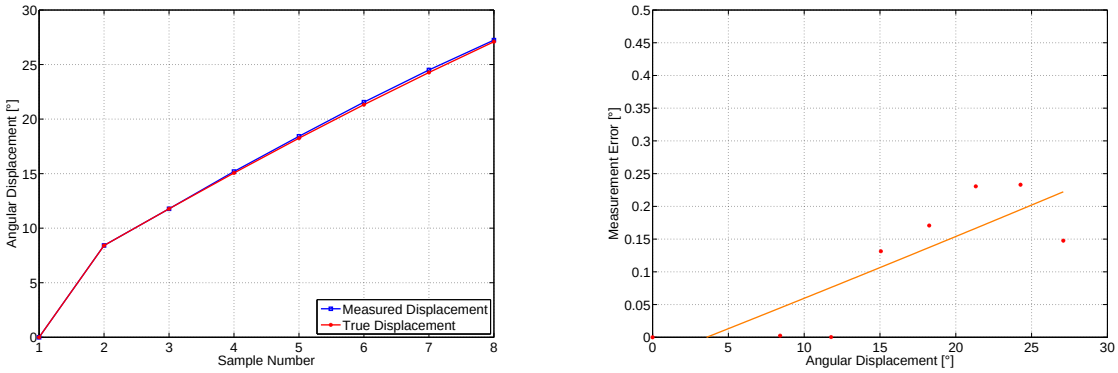
Notable observations regarding the full-scale translation tests include:

1. The VPAM node has the desired functionality with regards to translation measurements in a full-scale setup.

2. The error between the measured range and the true distance is also in the centimetre range (10 cm at 1 m from the target's centre). The error increases with increasing distance from the centre of the target which can also be attributed to projection deformation and non-ideal barrel-distortion correction.

Attitude Test

In this test the camera was rotated around the z-axis in a number of discrete steps to simulate a yaw manouvre. The true angle was determined by using a laser pointer mounted on the camera clamp and a vertical ruler mounted on the wall next to the camera. Using the distance between the camera and the distance measured on the ruler, the true angle was computed using trigonometry. A comparison of the true angle and the measured angle is shown in Figure 7.13a, while the measurement error is shown in Figure 7.13b.



(a) Angle comparison

(b) Measurement error with linear fit

Figure 7.13: Full-scale test: Attitude results

The following comments can be made regarding the results:

1. The VPAM node has the desired functionality with regards to yaw angle measurements in a full-scale setup.

2. The error between the measured angle and the true angle is very small – in the order of 0.15°. The increasing error at larger angles can be attributed to parallax errors (in the order of 3 mm) on the vertical ruler during the measurement process.

7.5 Summary

In this chapter practical aspects such as enclosure design, compensation for mounting offsets and barrel-distortion correction were discussed. The calibration procedure and the proposed modifications were discussed as well as the shortcomings and practical results of the technique. The final section outlined two different tests that were performed to confirm the functionality and accuracy. High-accuracy measurements were obtained which could be further improved through better calibration and improvement of non-circular marker projections.

Chapter 8

Conclusions and Recommendations

This chapter concludes the thesis with remarks about the research project and recommendations for related future work.

8.1 Conclusions

8.1.1 Computer Vision Algorithms

1. The matching algorithm developed in [92] was further analysed in this project. An empirical method to determine the matching threshold which limits the number of false rejections and invalid matches due to occlusions and phantom sources was developed and confirmed through simulation.

2. A pose estimation algorithm was developed to reduce the computational load and allow implementation on a small, low power device. The algorithm was compared to the iterative method [92] in terms of runtime, parameter sensitivity and robustness against noise. It is superior in all three cases. Simulation indicates position accuracy in the region of 2.5 cm and angular accuracy below 1°.

8.1.2 Camera Hardware

1. An integrated VPAM node capable of capturing high-contrast images and performing the processing on a low-power, low-cost processor was developed. Processing was done in a pipelined fashion to increase the framerate.

2. Increasing the image sensor clock frequency to 40 MHz was recommended by [92] to reduce the time required to download the image. Despite measures taken to reduce noise, the maximum clock frequency where the image transfer was reliable was 25 MHz.

3. To facilitate the pipelined architecture, the amount of data transferred from the image sensor was reduced using an RLE compression scheme. The decompression

was performed by a custom region growing algorithm.

4. Mass storage in the form of an SD card was added to the camera for data logging and debugging. A FAT16 file system was implemented to allow easy access to the data on any PC. In addition: a USB port was included in the design to allow faster image downloads during development and debugging.

5. The VPAM node was calibrated and its functionality was tested using a small-scale test configuration. The accuracy of the node was determined in a more practical situation using a full-scale test configuration. Position accuracy was in the low centimetre range, while angular accuracy was less than a degree which compares very well to existing VPAM systems.

8.1.3 Estimator

1. The simplified estimator developed during previous engineering projects in the ESL was documented and augmented to include visual measurements. The functionality was verified through simulation.

2. The effect of multiple sensors during measurement updates was investigated and a logic-based selection process was introduced to prevent one sensor from deteriorating the improvement caused by another sensor.

3. When there is a sensor domain transition, it can lead to sudden changes in state estimates. A smooth transition was introduced to prevent these sudden changes.

8.1.4 Control System

1. A waypoint-navigation guidance algorithm was developed capable of guiding the helicopter through a sequence of waypoints. The guidance parameters, such as waypoint positions, longitudinal velocity between waypoints and hover time at each waypoint, can be adjusted during execution using the groundstation software (Appendix B).

2. A landing guidance loop was developed which is capable of performing a normal landing utilising the improved state estimates from the augmented state estimator. This loop is independent of the method by which the measurements were obtained leaving the opportunity open for alternative vision or DGPS implementations.

3. The landing accuracy of the system is acceptable with 66.7% of the touchdowns within the desired region and the 31.4% in a slightly larger 2 m circular area. Only 1.9% of the landings was outside these regions.

8.1.5 Complete System

1. The standard HIL distribution board and Simulink interface was modified to accept 16 bytes of additional sensor data. The transfer rate between the computer and

the distribution board was increased to increase the time available to Simulink for performing model propagation.

2. A standard avionics system was built and thoroughly tested using the modified HIL system. The necessary brackets to mount the avionics and the VPAM node on the helicopter were designed and manufactured. A test flight to verify the functionality of the avionics was performed, but an RC receiver failure caused the swash plate to tilt forward aggresively which lead to a crash and severe damage to the airframe.

3. Due to time constraints, the system was only verified through simulation. The accuracy of the HIL system has been confirmed through previous projects at the ESL which leads to the conclusion that the landing system could be practically demonstrated with a high certainty of success with few modifications.

8.2 Recommendations

8.2.1 Computer Vision Algorithms

1. The TRIAD algorithm is often used in estimation to determine vehicle attitude from accelerometer and magnetometer measurements. A weighted TRIAD algorithm could be implemented to perform the pose estimation. This approach will take the relative magnitude of the high intensity regions into account which could improve accuracy.

8.2.2 Camera Hardware

1. It was recommended by [92] to reduce noise on the power supply and data lines to increase the clock frequency of the image sensor to above 20 MHz. Despite additional filters on the power supply, in-line buffers, separated analogue and digital ground planes and general low noise PCB layout techniques, the maximum attainable speed was limited to 25 MHz. The camera design was separated into a controller board and an image sensor board to keep the size down. This, however, requires a ribbon cable connection between the two boards which is not ideal for keeping noise levels under control. A more integrated design on a single PCB with the image sensor and FPGA on opposing sides to keep the connections short is advised. In this configuration care must be taken with the ground planes, especially with regard to the analogue circuitry of the image sensor.

2. High-power infra-red LEDs (developed by Roither Laser) typically used for scientific applications were identified but never purchased due to long lead times and high import costs. They have a wider beam angle than the units used during this project and will increase the operating envelope of the system. It is recommended that these units are acquired for future implementation and practical testing.

3. The entire image is processed during each iteration. An even higher frame rate is possible if the region-of-interest concept (developed in [92]) is reintroduced. Only

reading and processing regions of interest dramatically increases the frame rate, but is potentially not as robust since phantom centroids from previous frames could be propagated to new frames. At the same time, if the first identification of a region of interest does not include phantom centroids, it could be more robust since phantom centroids outside the regions of interest will not influence future frames. To maintain the current robustness of the system, a full investigation into the region of interest concept is required before it is reintroduced.

4. A helicopter is a harsh environment for electronic systems due to vibrations caused by the engine and blade assembly. A thorough investigation into the exact spectral response of the helicopter is recommended to assist in the design of a vibration reduction solution. If there are excessive high frequency vibrations when the image is captured, the image could be blurred and affect the accuracy of the pose estimation.

5. A single set of target markers is used to determine position and attitude. These markers must be well separated to be positively identified from the air which causes them to disappear from the field of view at around 1 m above the platform. GPS and the inertial sensors are then used in the last phase of the descent which results in a number of landings not touching down within the specified bounds. If a second set of markers with smaller spacing is placed within the first set, they could be used to maintain the high accuracy estimates until touchdown. The correspondence matching algorithm must then be adapted to compensate for the additional set of markers.

6. The vision-based position- and attitude-measurement algorithms are sensitive to deviations in the shape of the marker projections. Techniques to create more circular projections should be investigated.

7. The aluminium plate used for camera calibration should be modified to allow a full-frame image to be used in the calibration process.

8.2.3 Estimator

1. A system was implemented to use the last obtained vision measurement to correct the GPS measurements for a certain period after a sensor domain transition has occurred. This system did not produce satisfactory results and the cause must be investigated since it will be very advantageous during the final phase of the descent when the target markers have disappeared from the field of view.

8.2.4 Control System

1. The main focus of this project was to develop the necessary hardware to sufficiently improve the state estimates to allow for autonomous landing. The controllers outlined in Chapter 4 were developed to demonstrate the system but are not optimal. Aerodynamic effects such as the ground effect must be analysed in more detail to determine safe operating envelopes.

8.3 Project Summary

In this project the subsystems required to perform autonomous helicopter landing in z circular region with a 1 m radius were investigated and developed. Results obtained through simulation leads to the conclusion that the landing system could be practically demonstrated with a high certainty of success with few modifications.

Bibliography

[1] Gartner Dataquest. [cited at p. 78]

[2] Federal Aviation Administration. *Rotorcraft Flying Handbook*. U.S. Department of Transportation, 2000. [cited at p. 11, 12]

[3] Altera. *Cyclone II Device Handbook*. Altera, 2008. [cited at p. 78]

[4] E. Altug and C. Taylor. Vision-based pose estimation and control of a model helicopter. In *Proceedings of the IEEE International Conference on Mechatronics*, pages 316–321, 3-5 June 2004 2004. [cited at p. 5]

[5] T. G. Amaral, V. F. Pires, and M. M. Crisostomo. Autonomous landing of an unmanned aerial vehicle. In *CIMCA '05: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-2 (CIMCA-IAWTIC'06)*, pages 530–536, Washington, DC, USA, 2005. IEEE Computer Society. [cited at p. 1]

[6] Omead Amidi. *An Autonomous Vision-Guided Helicopter*. PhD thesis, Carnegie Mellon University, 1996. [cited at p. 1, 9]

[7] Jasbir Arora. *Introduction to Optimum Design*. Academic Press, 2 edition, May 2004. [cited at p. 37, 38]

[8] K.S. Arun, T.S. Huang, and S.D. Blostein. Least squares fitting of two 3D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698–700, September 1987. [cited at p. 40, 44, 45]

[9] Eric Albert Baker. The design of a CMOS sensor camera system for a nanosatellite. Master's thesis, Stellenbosch University, 2006. [cited at p. 78]

[10] Nicol Carstens. Development of a low-cost, low-weight flight control system for an electrically powered model helicopter. Master's thesis, Stellenbosch University, 2005. [cited at p. 9]

[11] A. Cesetti, E.Frontoni, A. Mancini, P. Zingaretti, and S. Longhi. A single camera feature-based vision system for helicopter autonomous landing. In *Proceedings of the International Conference on Advanced Robotics*, Munich, Germany, 22-26 June 2009. [cited at p. 6]

[12] Sierra Nevada Corporation. UAV common automatic recovery system version 2 for shipboard operations. In *Product Sheet*, 2010. [cited at p. 7]

[13] Cypress. CY7C1069DV33 datasheet. Technical report, Cypress, 2007. [cited at p. 81]

[14] Cypress. IBIS5 datasheet. Technical report, Cypress, 2009. [cited at p. 78, 113]

[15] N. Daucher, M. Dhome, J.T. La Preste, and G. Rives. Modelled object pose estimation and tracking by monocular vision. pages 249–258, 1993. [cited at p. 34, 40]

[16] E.R. Davies. *Machine Vision: Theory, Algorithms and Practicalities*. Morgan Kaufmann, 2005. [cited at p. 19, 24, 86]

[17] A.M. de Jager and I.K. Peddle. The design and implementation of a redundant inertial measurement unit. *Proceedings of European Air and Space Conference 2009 (Accepted)*, 2009. [cited at p. 11]

[18] Jaime Del-Cerro, Antonio Barrientos, Pascual Campoy, and Pedro J. Garcia. An autonomous helicopter guided by computer vision for inspection of overhead power cable. In *Proceedings of IEEE/RSJ International Conference of Intelligent Robots and Systems*, pages 69–78, Lausanne, Switzerland, 2002. [cited at p. 9]

[19] John E. Dennis and Robert B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Society for Industrial Mathematics, 1987. [cited at p. 37]

[20] Jason Denton and J. Ross Beveridge. Two dimensional projective point matching. In *SSIAI '02: Proceedings of the Fifth IEEE Southwest Symposium on Image Analysis and Interpretation*, page 77, Washington, DC, USA, 2002. IEEE Computer Society. [cited at p. 23]

[21] Electus Distribution. Understanding and using ccd cameras. Technical report, Electus Distribution, 2001. [cited at p. 108]

[22] Z. Engin, M. Lim, and A. Bharath. Gradient field correlation for keypoint correspondence. In *IEEE International Conference on Image Processing*, volume 2, pages 481–484, San Antonio, TX, Sept. 16 2007-Oct. 19 2007 2007. [cited at p. 23, 24]

[23] Michel Dhome Jean-Marc Lavest Eric Royer, Maxime Lhuillier. Performance evaluation of a localization system relying on monocular vision and natural landmarks. [cited at p. 50]

[24] S. M. Esmailifar and F. Saghafi. Autonomous unmanned helicopter landing system design for safe touchdown on 6DOF moving platform. *Autonomic and Autonomous Systems, International Conference on*, 0:245–250, 2009. [cited at p. 7]

[25] J. Fabrizio and J. Devars. An analytical solution to the perspective-n-point problem for common planar camera and for catadioptric sensor. *International Journal of Image and Graphics*, 8(1):135–155, January 2008. [cited at p. 40, 41, 43]

[26] M. Nasir Sulaiman-Mohamed H Shaharom Hariyati S. A. Majid Fakhrul Yusoff, Rahmita Rahmat. Establishing the straightness of a line for radial distortion correction through conic fitting. *IJCSNS International Journal of Computer Science and Network Security*, 9, 2009. [cited at p. 49, 106]

[27] Franklin and Powerll. *Dynamic Control System Design*. [cited at p. 55, 56, 57]

[28] D. Merchant G. Seedahmed, T. Schenk. Experimental results of digital camera calibration. *International Society for Photogrammetry and Remote Sensing*, 32:91–96, 1998. [cited at p. 49, 106]

[29] Pedro J. Garcia-Pardo, Gaurav S. Sukhatme, and James F. Montgomery. Towards vision-based safe landing for an autonomous helicopter. *Robotics and Autonomous Systems*, 38(1):19 – 29, 2002. [cited at p. 6, 7]

[30] Matt Garratt, Hemanshu Pota, Sebastien Eckersley-Maslin Andrew Lambert, and Clement Farabet. Visual tracking and LIDAR relative positioning for automated launch and recovery of an unmanned rotorcraft from ships at sea. In *Proceedings of the American Society for Naval Engineers Launch and Recovery Conference*, Annapolis, USA., 19-21 May 2008. [cited at p. viii, 7, 8]

[31] Stephanus Groenewald. Development of a rotary-wing test bed for autonomous flight. Master's thesis, Stellenbosch University, 2005. [cited at p. 9, 10, 93, 94]

[32] Gaurav Gupta. Autonomous landing of UAV using vision. Manuscript written and submitted on 21 November 2005 as course work for EE672: Computer Vision and Document Processing under guidance of Dr RMK Sinja, Prof. Department of Electrical Engineering and Computer Science and Engineering, IIT Kanpur, India., November 2005. [cited at p. 2]

[33] R. Sepponen H. Rimminen, J. Lindstrm. Positioning accuracy and multi-target separation with a human tracking system using near field imaging. *INTERNATIONAL JOURNAL ON SMART SENSING AND INTELLIGENT SYSTEMS*, 2, 2009. [cited at p. 50]

[34] Janne Heikkila. Accurate camera calibration and feature based 3-d reconstruction from monocular image sequences. [cited at p. 107]

[35] Jeffrey Ho, Ming-Hsuan Yang, Anand Rangarajan, and Baba Vemuri. A new affine registration algorithm for matching 2D point sets. In *WACV '07: Proceedings of the Eighth IEEE Workshop on Applications of Computer Vision*, page 25, Washington, DC, USA, 2007. IEEE Computer Society. [cited at p. 23]

[36] B.K.P. Horn. Closed form solution of absolute orientation using unit quarternions. *Journal of the Optical Society of America*, 5(7):1127–1135, 1987. [cited at p. 40]

[37] Willem Hough. Autonomous aerobatic flight of a fixed wing unmanned aerial vehicle. Master's thesis, Stellenbosch University, December 2008. [cited at p. ix, 10, 19, 52, 56, 59, 60, 93, 96]

[38] Z. Y. Hu and F. C. Wu. A note on the number of solutions of the noncoplanar P4P problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):550–555, 2002. [cited at p. 34]

[39] Andrew Johnson and Jim Montgomery. Vision guided landing of an autonomous helicopter in hazardous terrain. 2005. [cited at p. 6, 7]

[40] Stephen J. Wright Jorge Nocedal. *Numerical optimization.* 1999. [cited at p. 36, 37, 38]

[41] I.K. Jung and S. Lacroix. A robust interest points matching algorithm. *Proceedings of the Eighth IEEE International Conference on Computer Vision*, 2:538–543, 2001. [cited at p. 23, 24]

[42] Michael Bosse Karl, W. C. Karl, and Paul Debitetto. A vision augmented navigation system. In *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, pages 1028–33, 1997. [cited at p. 1]

[43] Jonathan Kelly, Srikanth Saripalli, and Gaurav S. Sukhatme. Combined visual and inertial navigation for an unmanned aerial vehicle. *Field and Service Robotics*, 42:255–264, 2008. [cited at p. 9]

[44] Y. Kudo, T. Kajiyama, and W. Mitsuhashi. Extraction of feature points suitable for matching image correspondence. In *SICE 2003 Annual Conference*, volume 1, pages 26 – 29, 4-6 Aug. 2003 2003. [cited at p. 24]

[45] A Isidori L Marconi and A Serrani. Autonomous vertical landing on an oscillating platform: an internal model based approach. *Automatica*, 38, 2002. [cited at p. 7]

[46] Ping Li, Dirk Farin, Rene Gunnewiek, and Peter de With. Contrast-invariant feature point correspondence. In *Proceedings of IEEE International Conference on Speech and Signal Processing*, volume 1, pages 477–480, Honolulu, 15-20 April 2007 2007. [cited at p. 23, 24]

[47] M. L. Liu and K. H. Wong. Pose estimation using four corresponding points. *Pattern Recognition Letters*, 20(1):69–74, 1999. [cited at p. 34]

[48] Manolis I. A. Lourakis. A brief description of the Levenberg-Marquardt algorithm implemented by levmar. 2005. [cited at p. 38]

[49] David G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:441–450, 1991. [cited at p. 34]

[50] Pei Luo and Hai long Pei. An autonomous helicopter with vision based navigation. In *Proceedings of IEEE International Conference on Control and Automation*, Guangzhou, China, June 2007 2007. [cited at p. 9]

[51] K. Madsen, H.B. Nielsen, and O. Tingleff. Methods for non-linear least squares problems, 2004. [cited at p. 35, 37, 38]

[52] Daniel Francois Malan. 3D tracking between satellites using monocular computer vision. Master's thesis, University of Stellenbosch, December 2004. [cited at p. 19, 40]

[53] Luis Mejas and Pascual Campoy. Two seconds to touchdown. vision-based controlled forced landing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3527–3532, 2006. [cited at p. 6]

[54] Luis Mejias, Srikanth Saripalli, Pascual Campoy, and Gaurav Sukhatme. Visual servoing of an autonomous helicopter in urban areas using feature tracking. *Journal of Field Robotics*, 23:185–199, 2006. [cited at p. 6, 9]

[55] Katsuhiro Wakabayashi Shinya Yokoyama Ming Li, Kenji Imou. Review of research on agricultural vehicle autonomous guidance. *Int J Agric & Biol Eng*, 2, 2009. [cited at p. 50]

[56] Dewald Minie. Autonomous docking for a satellite pair using monocular vision. Master's thesis, Stellenbosch University, 2009. [cited at p. 106, 107]

[57] Paul Y. Montgomery, David G. Lawrence, Kurt. R. Zimmerman, H. Stewart Cobb, Gregory M. Gutt, and Clark E. Cohen. UAV application of IBLS for autonomous takeoff and landing. In *Proceedings of the 12th International Technical Meeting of the Satellite Division of The Institute of Navigation*, Nashville, TN, USA, 14-17 September 1999 1999. [cited at p. 7]

[58] Donald A. Neamen. *Electronic Circuit Analysis and Design.* [cited at p. 48]

[59] S. Nichani. Solving the correspondence problem using a Hopfield network. In *IEEE International Conference on Neural Networks*, volume 6, pages 4107 – 4112, Orlando, Florida, United States, 27 Jun- 2 Jul 1994 1994. [cited at p. 23, 24]

[60] Novatel. Novatel OEMV-1G receiver brochure. Technical report, Novatel, 2010. [cited at p. 2]

[61] So-Ryeok Oh, Kaustubh Pathak, Sunil K. Agrawal, Hemanshu R. Pota, and Matt Garratt. Autonomous helicopter landing on a moving platform using a tether. In *IEEE International Conference on Robotics and Automation*, pages 3960–3965, Barcelona, Spain, 18-22 April 2005. [cited at p. 7, 9]

[62] S.H. Or, W.S. Luk, and K.H. Wong. An efficient iterative pose estimation algorithm. 1997. [cited at p. 34, 40]

[63] Iain K. Peddle. Autonomous flight of a model aircraft. Master's thesis, Stellenbosch University, 2005. [cited at p. 97]

[64] Jian-Xun Peng, James Niblock, and Karen McMenemy. An iterative algorithm for finding point correspondences. In *CISP '08: Proceedings of the 2008 Congress on Image and Signal Processing, Vol. 1*, pages 441–447, Washington, DC, USA, 2008. IEEE Computer Society. [cited at p. 23]

[65] Alison A. Proctor and Eric N Johnson. Vision-only approach and landing. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, San Francisco, California, 15-18 August 2005 2005. American Institute of Aeronautics and Astronautics. [cited at p. 2, 3, 17, 24]

[66] Long Quan and Zhongdan Lan. Linear n-point camera pose determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:774–780, 1999. [cited at p. 34, 40]

[67] Fabio Remondino. Digital camera calibration methods: Considerations and comparisons. In *ISPRS Symposium: Image Engineering and Vision Metrology*, 2006. [cited at p. 107]

[68] Jan-Cor Roos. Autonomous take-off and landing of a fixed wing unmanned aerial vehicle. Master's thesis, Stellenbosch University, 2007. [cited at p. 97]

[69] Louis Emile Rossouw. Autonomous flight of an unmanned helicopter. Master's thesis, Stellenbosch University, 2008. [cited at p. 9, 10, 53, 65, 72, 94]

[70] Srikanth Saripalli and Gaurav Sukhatme. Landing a helicopter on a moving target. [cited at p. 1, 9]

[71] Srikanth Saripalli and Gaurav Sukhatme. Landing on a moving target using an autonomous helicopter. In *Proceedings of the International Conference on Field and Service Robotics*, 2003. [cited at p. 1, 7]

[72] Srikanth Saripalli, Gaurav S. Sukhatme, and James F. Montgomery. An experimental study of the autonomous helicopter landing problem. In *Proceedings, International Symposium on Experimental Robotics, (SantAngelo dIschia)*, pages 466–475, 2002. [cited at p. 1, 101]

[73] Gerald Schweighofer and Axel Pinz. Robust pose estimation from a planar target. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:2024–2030, 2006. [cited at p. 34]

[74] Robert Sedgewick. *Algorithms in Java*. Pearson Education, 2003. [cited at p. 26]

[75] Pedro Serra, Rita Cunha, and Carlos Silvestre. On the design of rotorcraft landing controllers. In *Proceedings of 16th Mediterranean Conference on Control and Automation*, 2008. [cited at p. 1]

[76] Omid Shakernia, Yi Ma, T. John Koo, and Shankar Sastry. Landing an unmanned air vehicle: Vision based motion estimation and non-linear control. *Asian Journal of Control*, 1:128–145, 1999. [cited at p. 1]

[77] Linda G. Shapiro and George C. Stockman. *Computer Vision*. Prentice Hall, 2001. [cited at p. 19, 23, 24, 86]

[78] Courtney Sharp, Omid Shakernia, and Shanker Sastry. A vision system for landing and unmanned aerial vehicle. *Proceedings of the 2001 IEEE Transactions on Robotics and Automation*, 2001. [cited at p. viii, 1, 2, 3, 4, 5, 17, 24]

[79] Hyunchul Shim. *Hierarchical Flight Control System Synthesis for Rotorcraft-based Unmanned Aerial Vehicles*. PhD thesis, University of California at Berkeley, 2000. [cited at p. 1]

[80] Bruno Sinopoli, Mario Micheli, Gianluca Donato, and John Koo. Vision based navigation for an unmanned aerial vehicle. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1757–1765, 2001. [cited at p. 1]

[81] Philip Smit. Development of a 3-DOF motion simulation platform. Master's thesis, Stellenbosch University, 2010. [cited at p. 10]

[82] G. S. Sukhatme Srikanth Saripalli, James F. Montgomery. Vision-based autonomous landing of an unmanned aerial vehicle. 2003. [cited at p. 1, 2, 6, 17]

[83] Gaurav S. Sukhatme Srikanth Saripalli, James F. Montgomery. Visually-guided landing of an unmanned aerial vehicle. *IEEE Transactions on Robotics and Automation*, 19(3):371–381, June 2003. [cited at p. viii, 1, 2, 6]

[84] J. G. Fryer T. A. Clarke. The development of camera calibration methods and models. [cited at p. 49, 106, 107]

[85] Ashit Talukder and David Casasent. Classification and pose estimation of objects using non-linear features. [cited at p. 40]

[86] Todd Templeton, David Hyunchul Shim, Christopher Geyer, and S. Shankar Sastry. Autonomous vision-based landing and terrain mapping using an MPC-controlled unmanned rotorcraft. In *IEEE International Conference on Robotics and Automation for consideration*, pages 1349–1356, 2006. [cited at p. 1, 6, 7]

[87] Todd R. Templeton. Autonomous vision-based rotorcraft landing and accurate aerial terrain mapping in an unknown environment. Master's thesis, University of California at Berkeley, 2007. [cited at p. 1]

[88] Gianpaolo Conte Torsten Merz, Simone Duranti. Autonomous landing of an unmanned helicopter based on vision and inertial sensing. [cited at p. viii, 2, 3, 4, 6, 17, 65]

[89] Bill Triggs. Camera pose and calibration from 4 or 5 known 3D points. In *Proceedings of the International Conference on Computer Vision*, pages 278–284. IEEE Computer Society Press, 1999. [cited at p. 34]

[90] Y Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf cameras and lenses. *IEEE Journal of Robotics and Automation*, 3:323–344, 1987. [cited at p. 107]

[91] Carlo van Schalkwyk. Full state control of a Fury X-Cell unmanned helicopter. Master's thesis, Stellenbosch University, 2009. [cited at p. 10]

[92] Bernardus Johannes Visser. The precision landing of a UAV. Master's thesis, Stellenbosch University, 2008. [cited at p. 10, 13, 16, 21, 34, 36, 38, 48, 53, 77, 78, 91, 97, 113, 118, 120]

[93] W.J. Wagtendonk. *Principles of Helicopter Flight*. Aviation Supplies & Academics Inc., second edition edition, 2006. [cited at p. 12, 70]

[94] M.D. Wheeler and Katsushi Ikeuchi. Iterative estimation of rotation and translation using the quaternion. 1995. [cited at p. 34]

[95] Allen D. Wu, Eric N. Johnson, and Alison A. Proctor. Vision-aided inertial navigation for flight control. In *Journal of Aerospace Computing, Information, and Communication*, 2005. [cited at p. 23, 24]

[96] Z. Xiong and Y. Zhang. A novel interest-point-matching algorithm for high-resolution satellite images. *GeoRS*, 47(12):4189–4200, December 2009. [cited at p. 23]

[97] Cui Xu, Liankui Qiu, Ming Liu, Bin Kong, and Yunjian Ge. Stereo vision based relative pose and motion estimation for unmanned helicopter landing. *International Conference on Information Acquisition*, 0:31–36, 2006. [cited at p. 7, 8]

[98] Zhenyu Yu, Kenzo Nonami, Jinok Shin, and Demian Celestino. 3D vision based landing control of a small scale autonomous helicopter. *International Journal of Advanced Robotic Systems*, 4:51–56, 2007. [cited at p. 6]

[99] Wei-Zhong Zhang, Qing-Guo Liu, and Li-Yan Zhang. Automatic image feature matching based on reference points. In *Proceedings of the International Conference on Machine Learning and Cybernetics*, Dalian, China, 13-16 August 2006 2006. [cited at p. 23, 24]

[100] Zhengyou Zhang. A flexible new technique for camera calib. Technical report, Microsoft Corporation, 1998. [cited at p. 107]

Appendices

Appendix A

CMOS Configuration

The following lookup tables are used during the CMOS configuration. ASCII values are used for the index to simplify communication from a terminal window.

Table A.1: Gain command lookup table

Gain Index	Real Gain	Gain Index	Real Gain
0x30 ('1')	1.37	0x39 ('9')	6.25
0x32 ('2')	1.62	0x61 ('a')	7.89
0x33 ('3')	1.96	0x62 ('b')	9.21
0x34 ('4')	2.33	0x63 ('c')	11
0x35 ('5')	2.76	0x64 ('d')	11.37
0x36 ('6')	3.5	0x65 ('e')	11.84
0x37 ('7')	4.25	0x66 ('f')	12.32
0x38 ('8')	5.2	0x67 ('g')	12.42

Table A.2: Integration time command lookup table

Integration Index	Time Command to Image Sensor	Integration Time
0x30 ('1')	79	1 ms
0x32 ('2')	157	2 ms
0x33 ('3')	314	4 ms
0x34 ('4')	626	8 ms
0x35 ('5')	1303	16 ms

Appendix B

Software

This appendix contains screen captures of the software used during the simulation of the systems developed in this project.

B.1 Groundstation

This section contains screen captures of the navigation (Figure B.1) and landing (Figure B.2) tabs developed for the groundstation software:

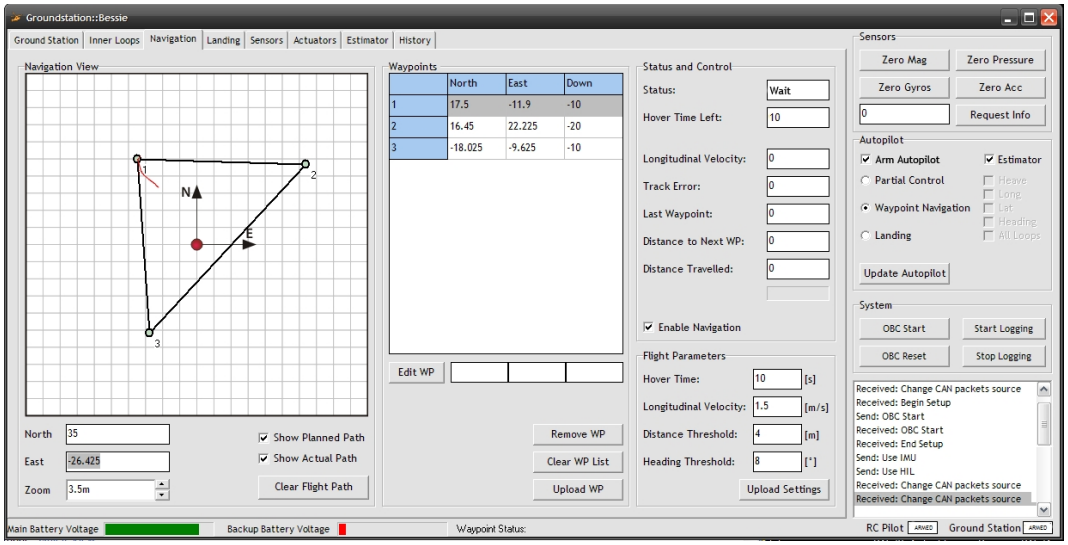


Figure B.1: The navigation tab in the groundstation software

B.2 Visualisation

The visualisation software shows a 3-D representation of the helicopter behaviour during software and HIL simulations. In Figure B.3 the helicopter is shown during the descent phase.

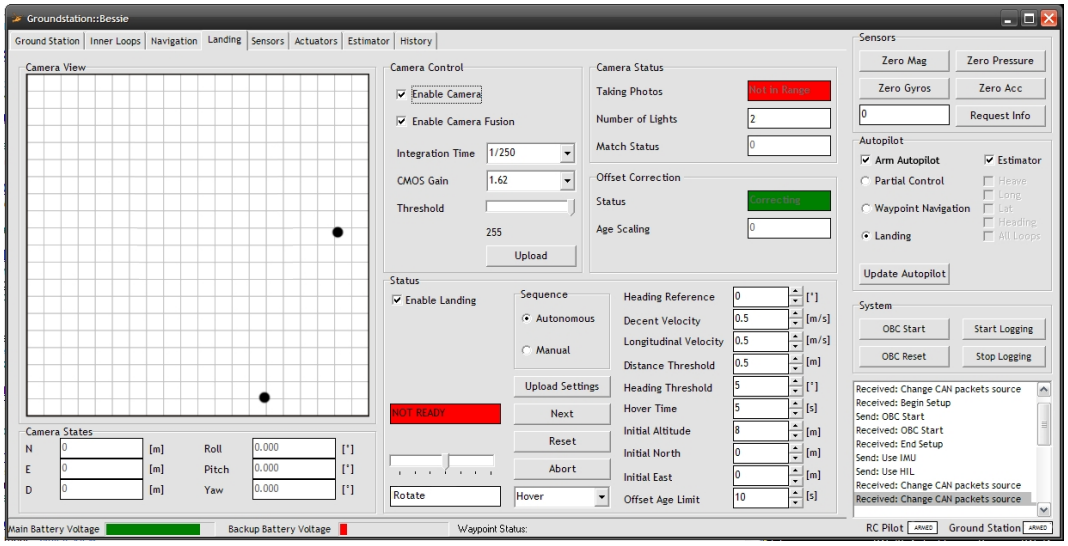


Figure B.2: The landing tab in the groundstation software

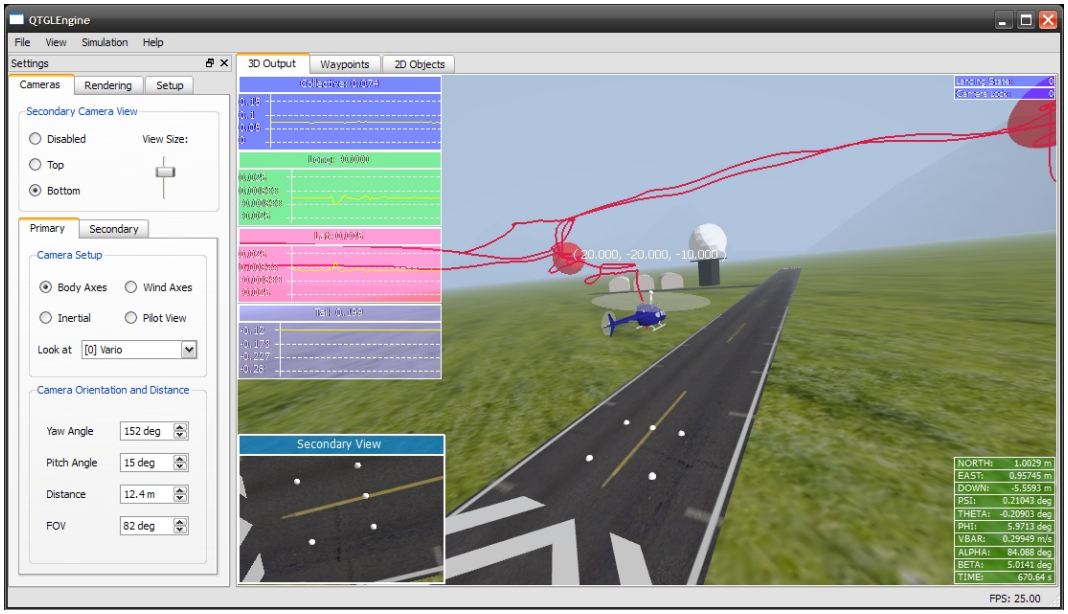


Figure B.3: The descent phase in the visualisation software during a HIL simulation

Appendix C

Additional Results

This appendix contains additional simulation results not shown in Chapters 3 and 6.

C.1 Estimator

This section contains the position, velocity and attitude states of the estimator simulation from Chapter 3.

C.1.1 Position and Velocity States

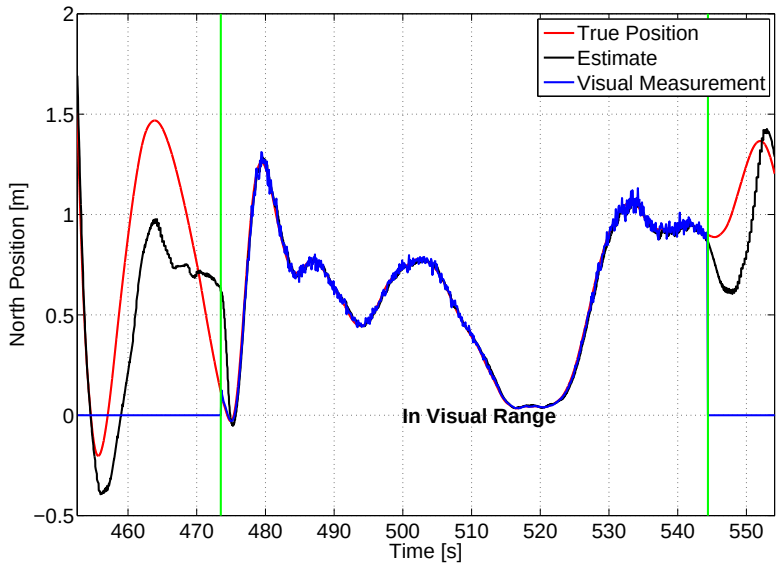


Figure C.1: Estimator results - North position

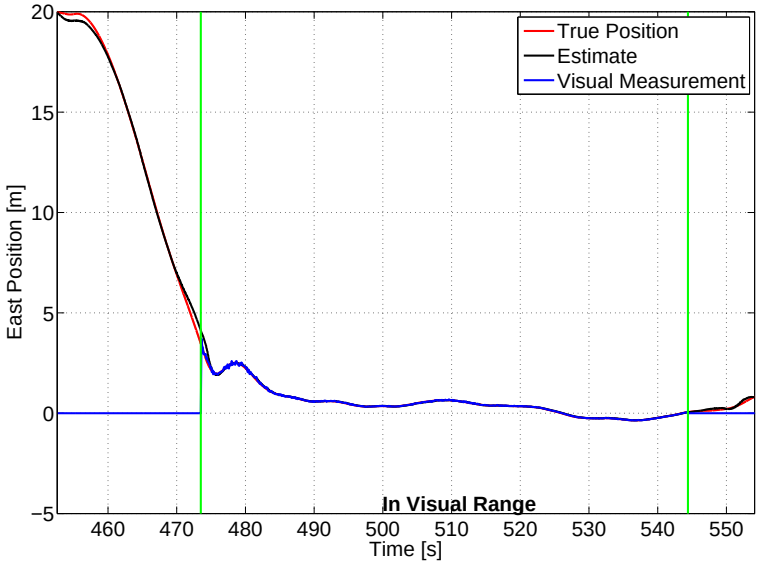


Figure C.2: Estimator results - East position

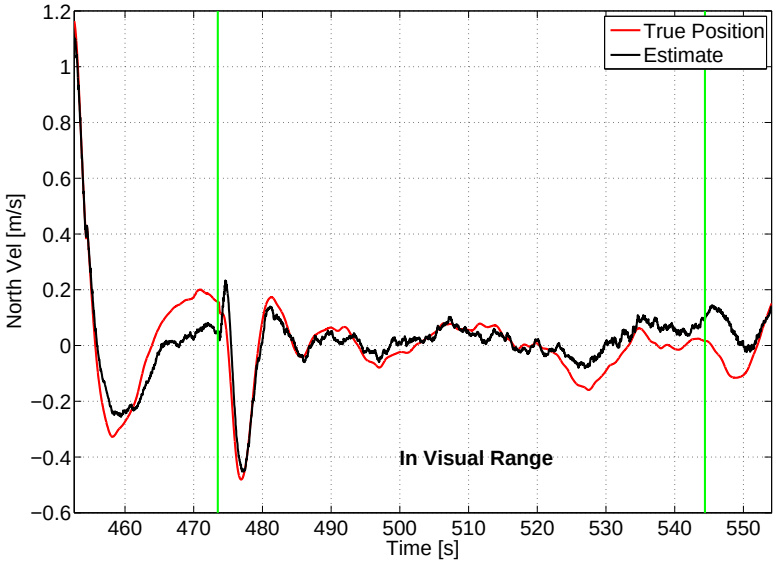


Figure C.3: Estimator results - North velocity

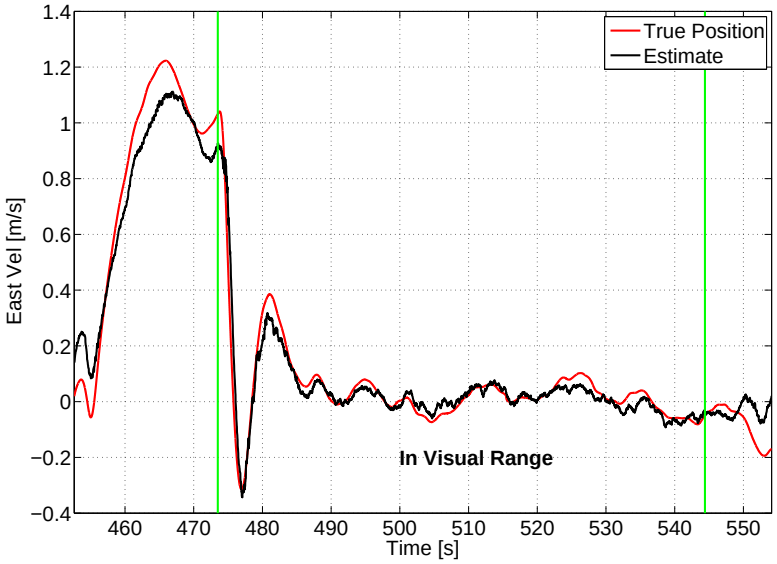


Figure C.4: Estimator results - East velocity

C.1.2 Attitude States

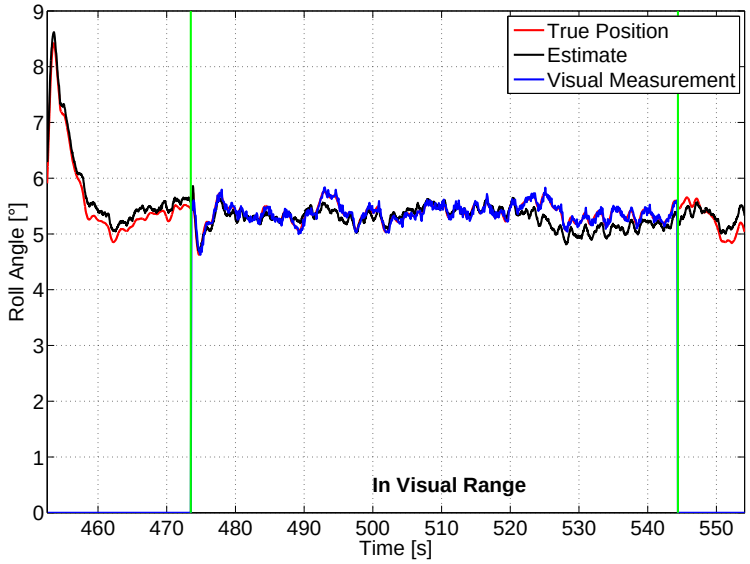


Figure C.5: Estimator results - Roll angle

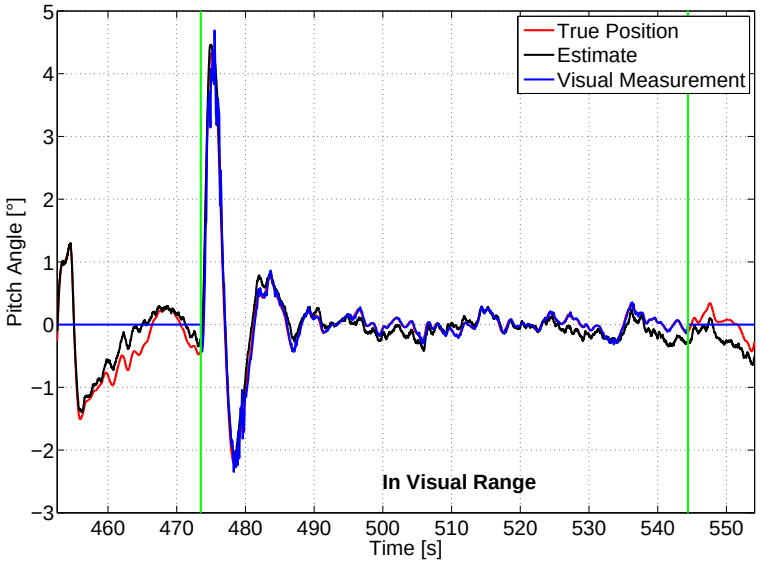


Figure C.6: Estimator results - Pitch angle

C.2 System Simulation

This section contains the state estimates from the system simulation in Chapter 6. The states during the waypoint navigation are shown in Section C.2.1 and the states during the landing procedure are shown in Section C.2.2.

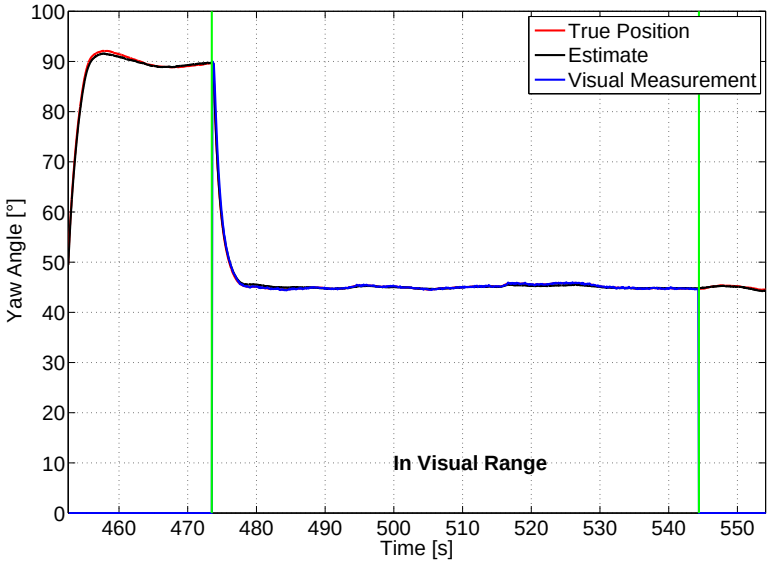


Figure C.7: Estimator results - Yaw angle

C.2.1 Waypoint Navigation

Position and Velocity States

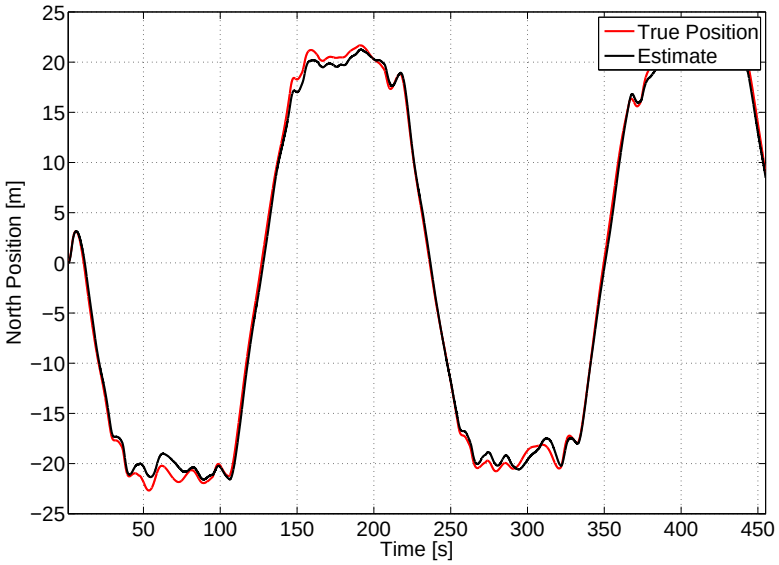


Figure C.8: Navigation results - North position

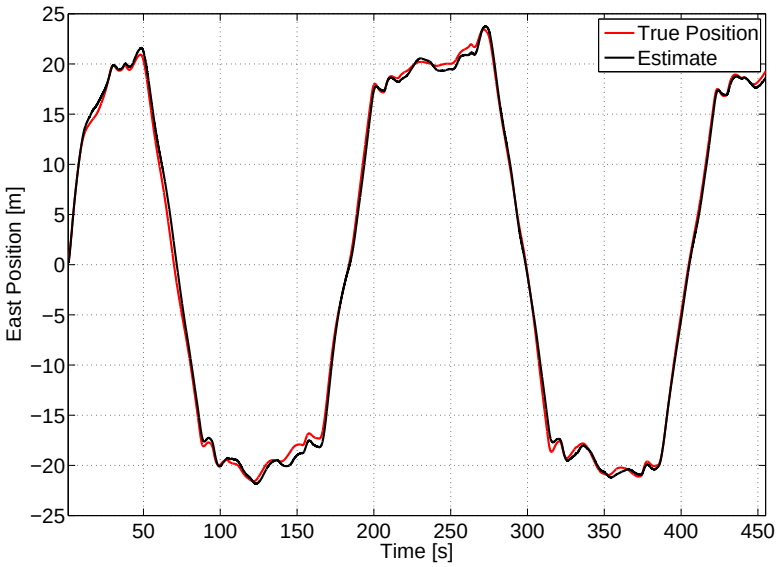


Figure C.9: Navigation results - East position

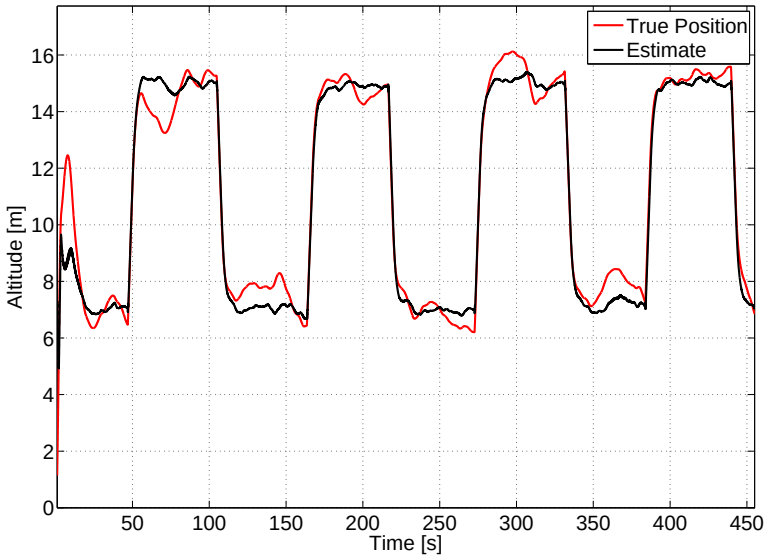


Figure C.10: Navigation results - Altitude

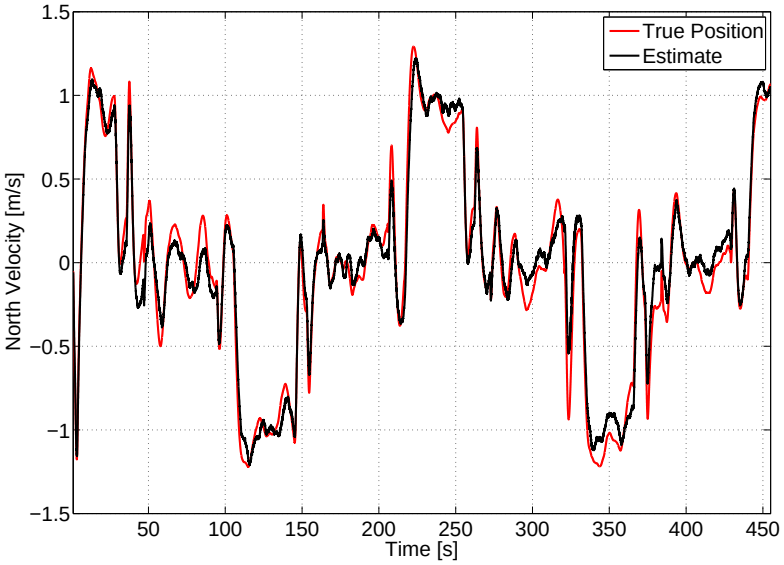


Figure C.11: Navigation results - North velocity

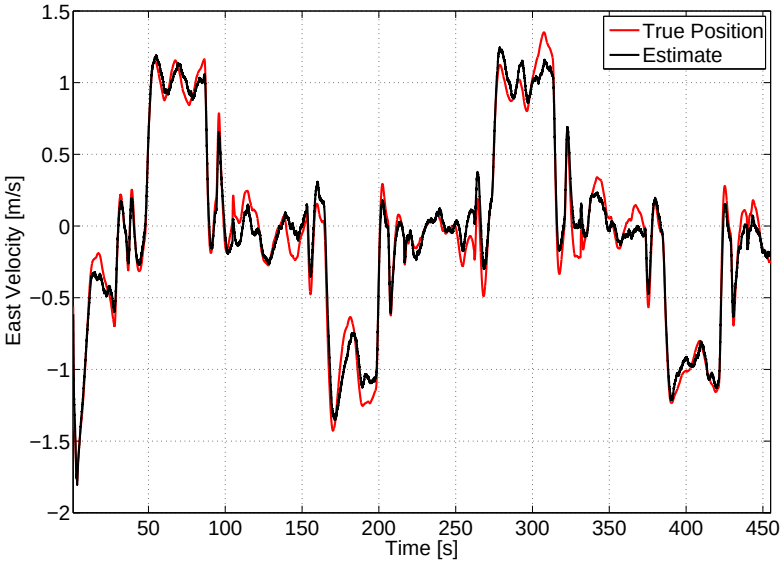


Figure C.12: Navigation results - East velocity

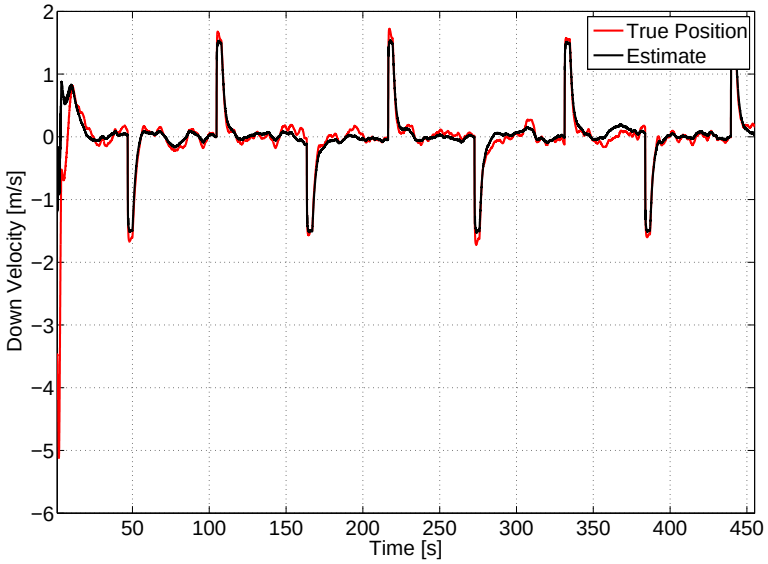


Figure C.13: Navigation results - Down velocity

Attitude States

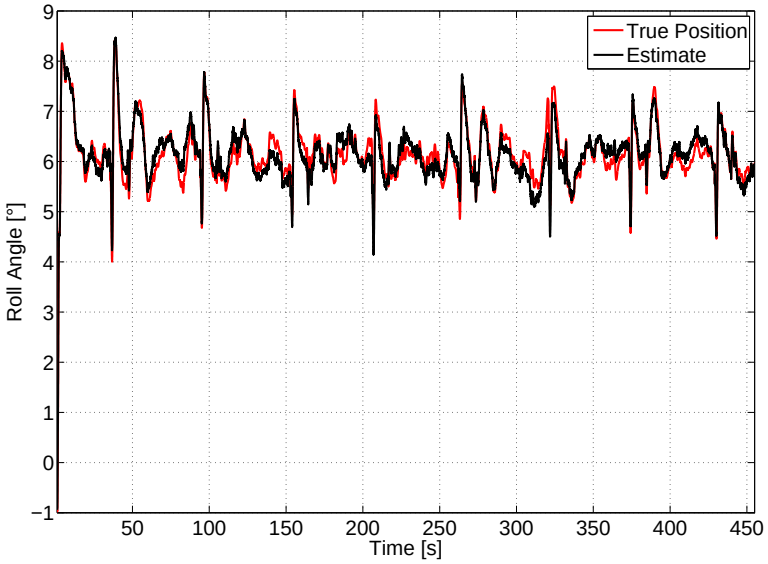


Figure C.14: Navigation results - Roll angle

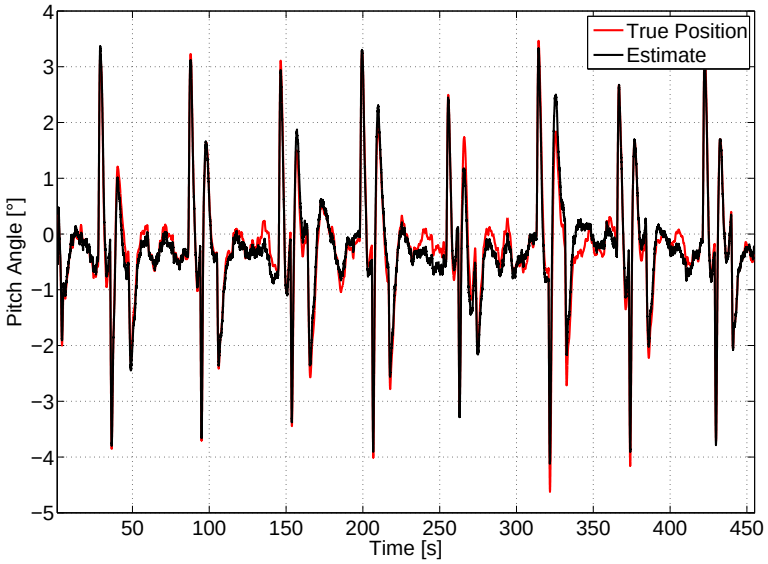


Figure C.15: Navigation results - Pitch angle

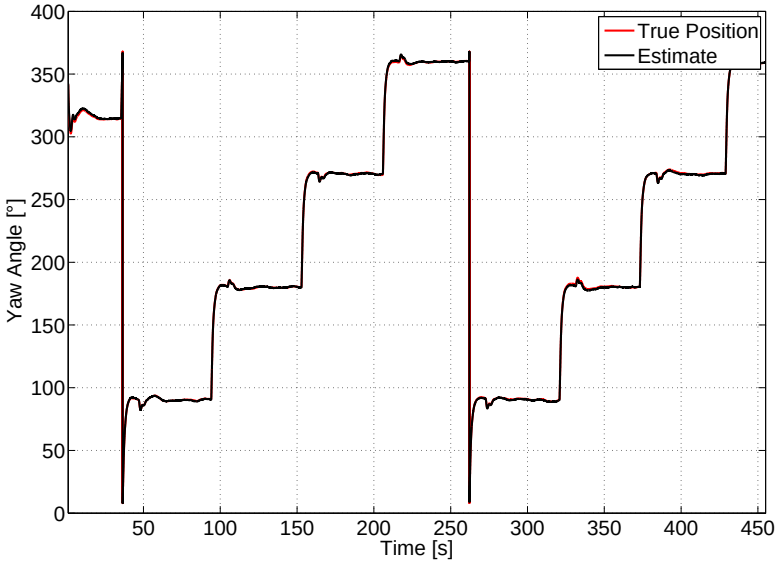


Figure C.16: Navigation results - Yaw angle

C.2.2 Landing

Position and Velocity States

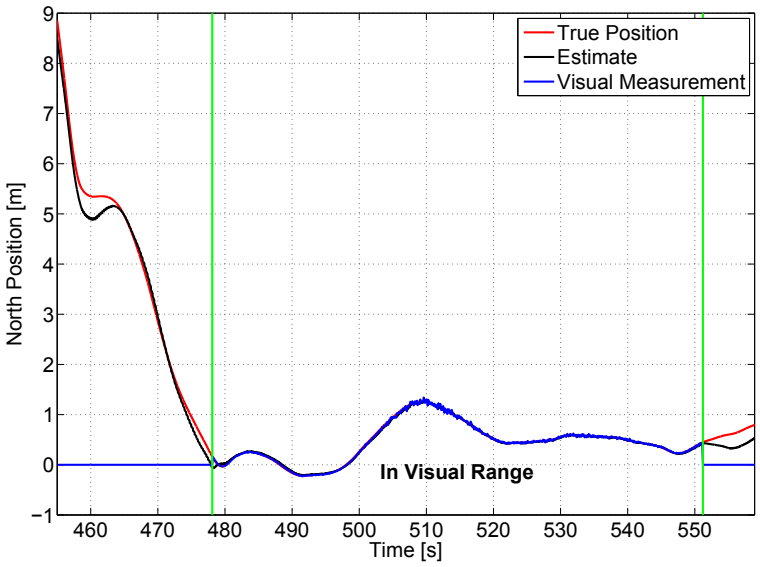


Figure C.17: Landing results - North position

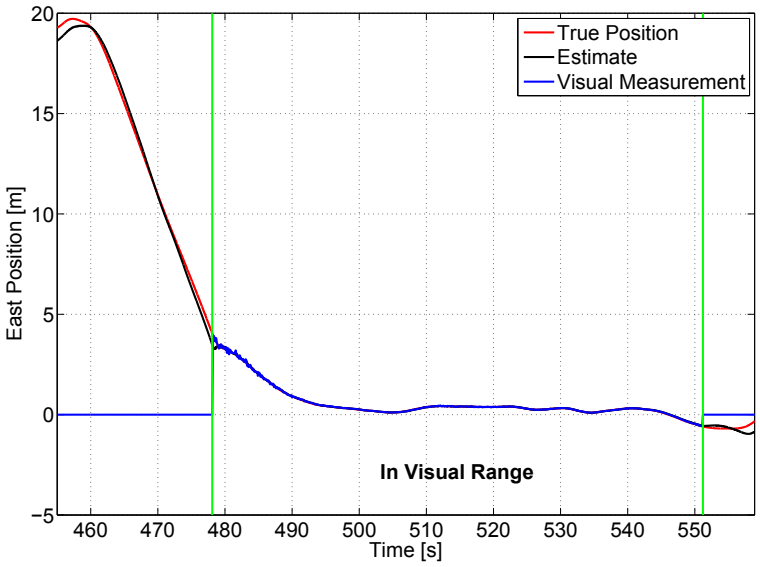


Figure C.18: Landing results - East position

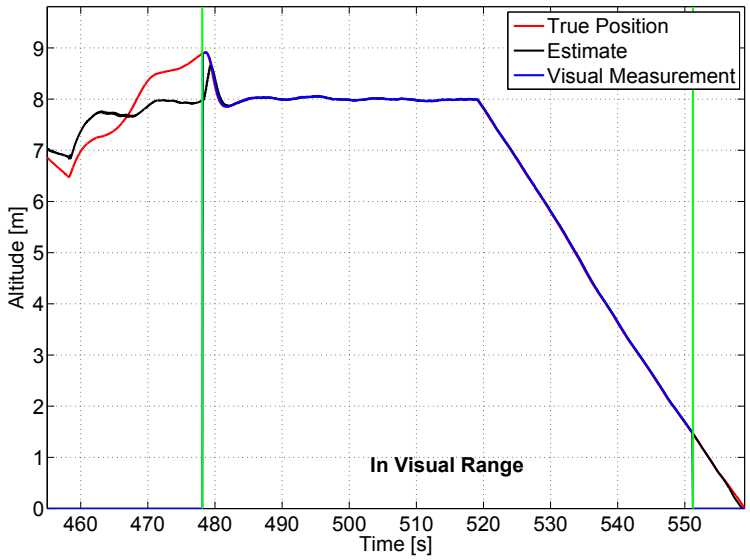


Figure C.19: Landing results - Altitude

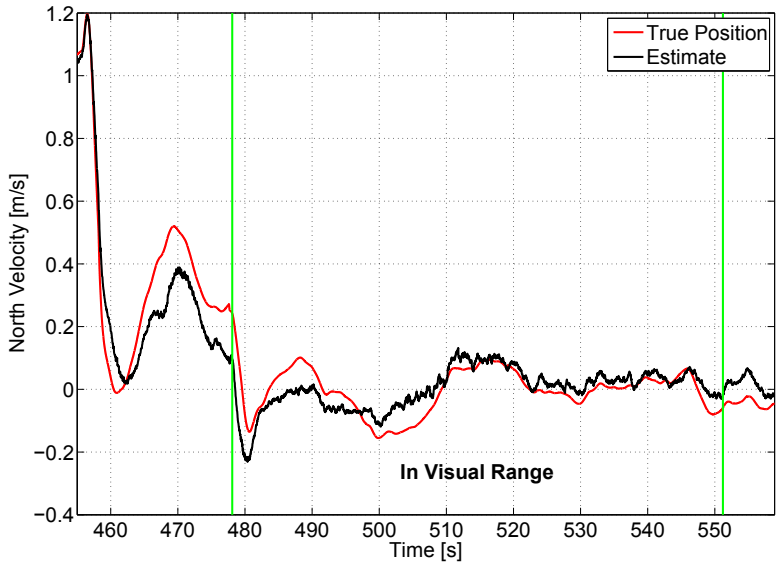


Figure C.20: Landing results - North velocity

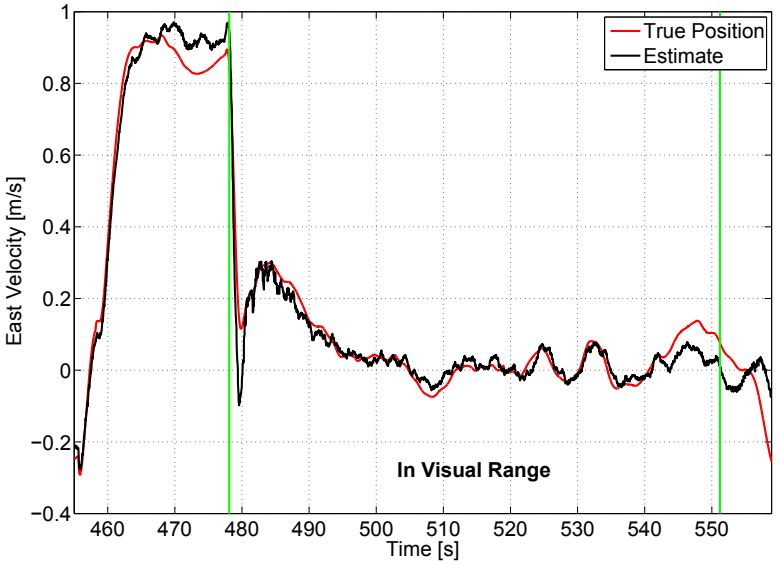


Figure C.21: Landing results - East velocity

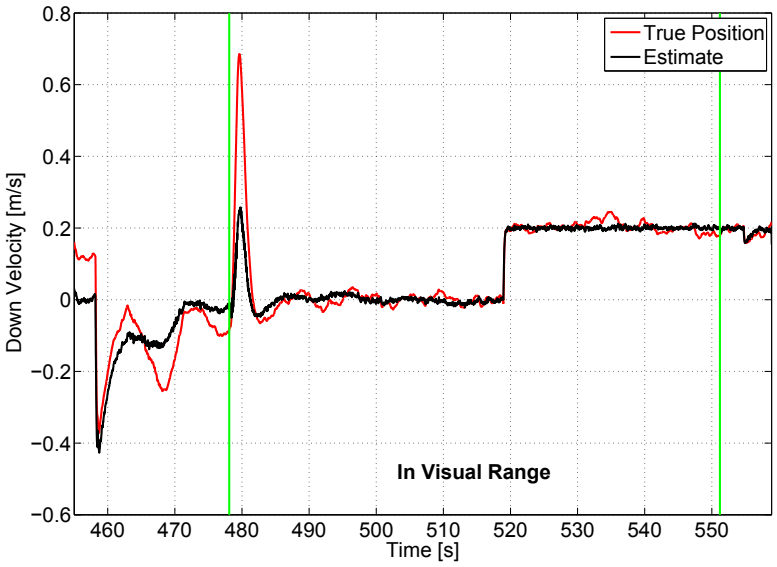


Figure C.22: Landing results - Down velocity

Attitude States

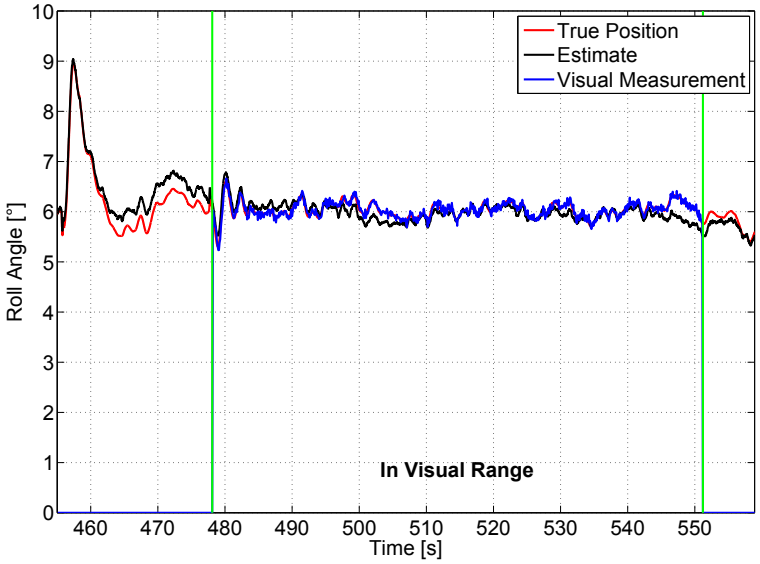


Figure C.23: Landing results - Roll angle

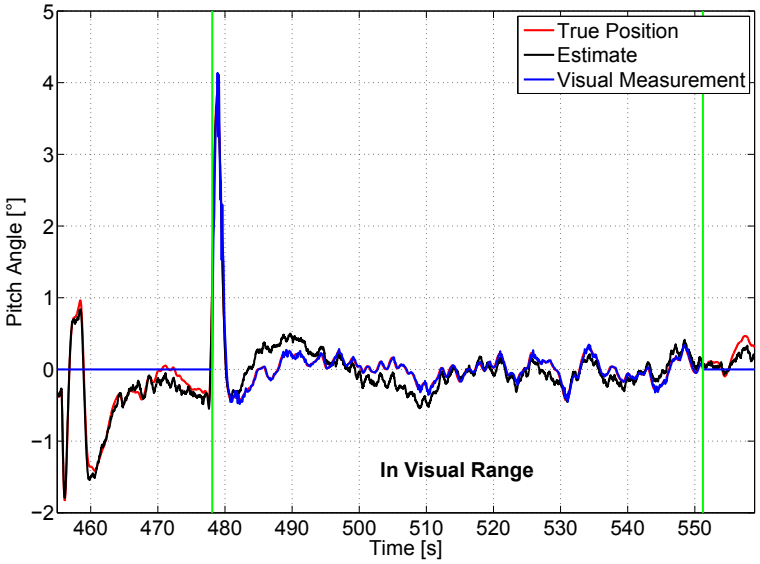


Figure C.24: Landing results - Pitch angle

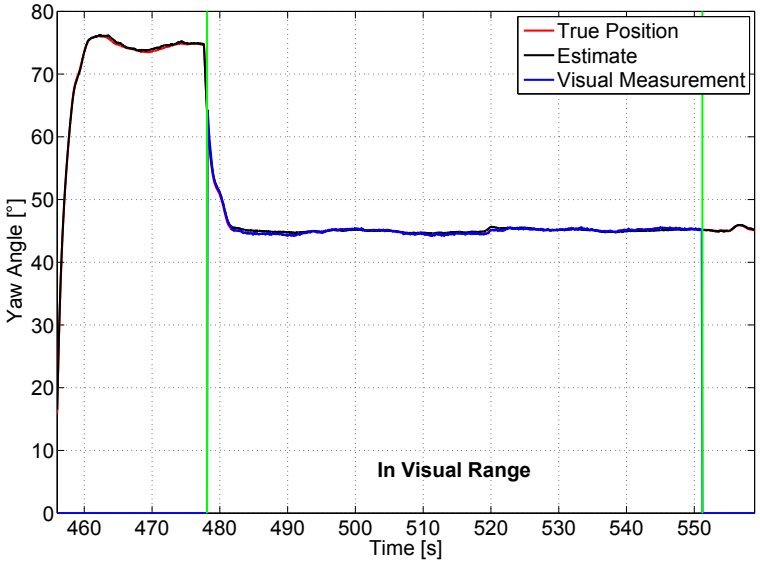


Figure C.25: Landing results - Yaw angle

Appendix D

Test Configurations

D.1 Calibration

The camera calibration was performed using an aluminium grid and diffused light to create the test pattern. This configuration is shown in Figure D.1:

Figure D.1: The calibration configuration

D.2 Small-Scale

A small-scale test configuration was used to test the functionality of the VPAM node during development. This configuration is shown in Figure D.2:

Figure D.2: The small-scale configuration

D.3 Full-Scale

A full-scale test configuration was used to confirm the functionality in a more practical environment as well as determine the system's accuracy. This configuration is shown in Figures D.3 and D.4:

Figure D.3: The full-scale test configuration: Side view

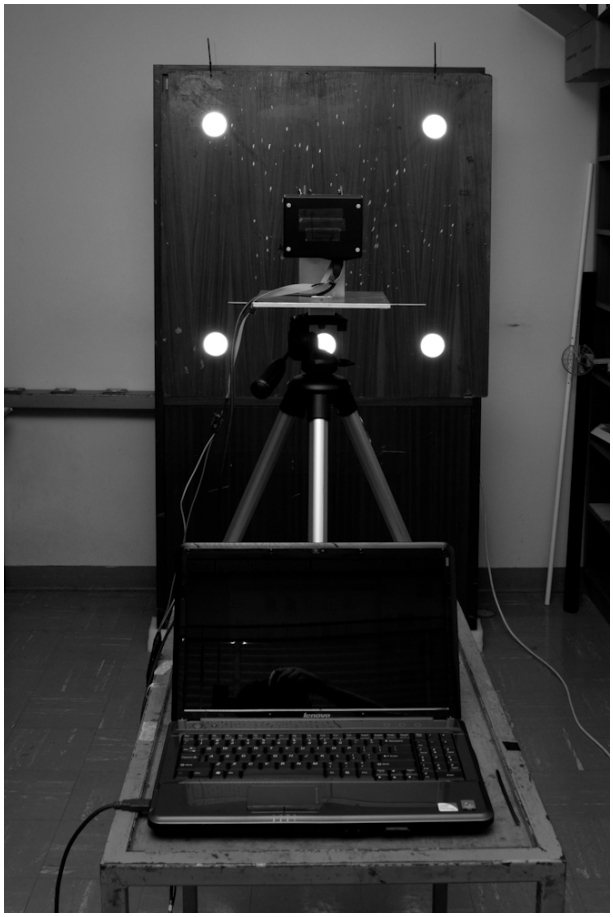


Figure D.4: The full-scale test configuration: Front view